ゲームプログラムでマスター
Quick BASIC

PC-9801対応

稲田浩一朗 著

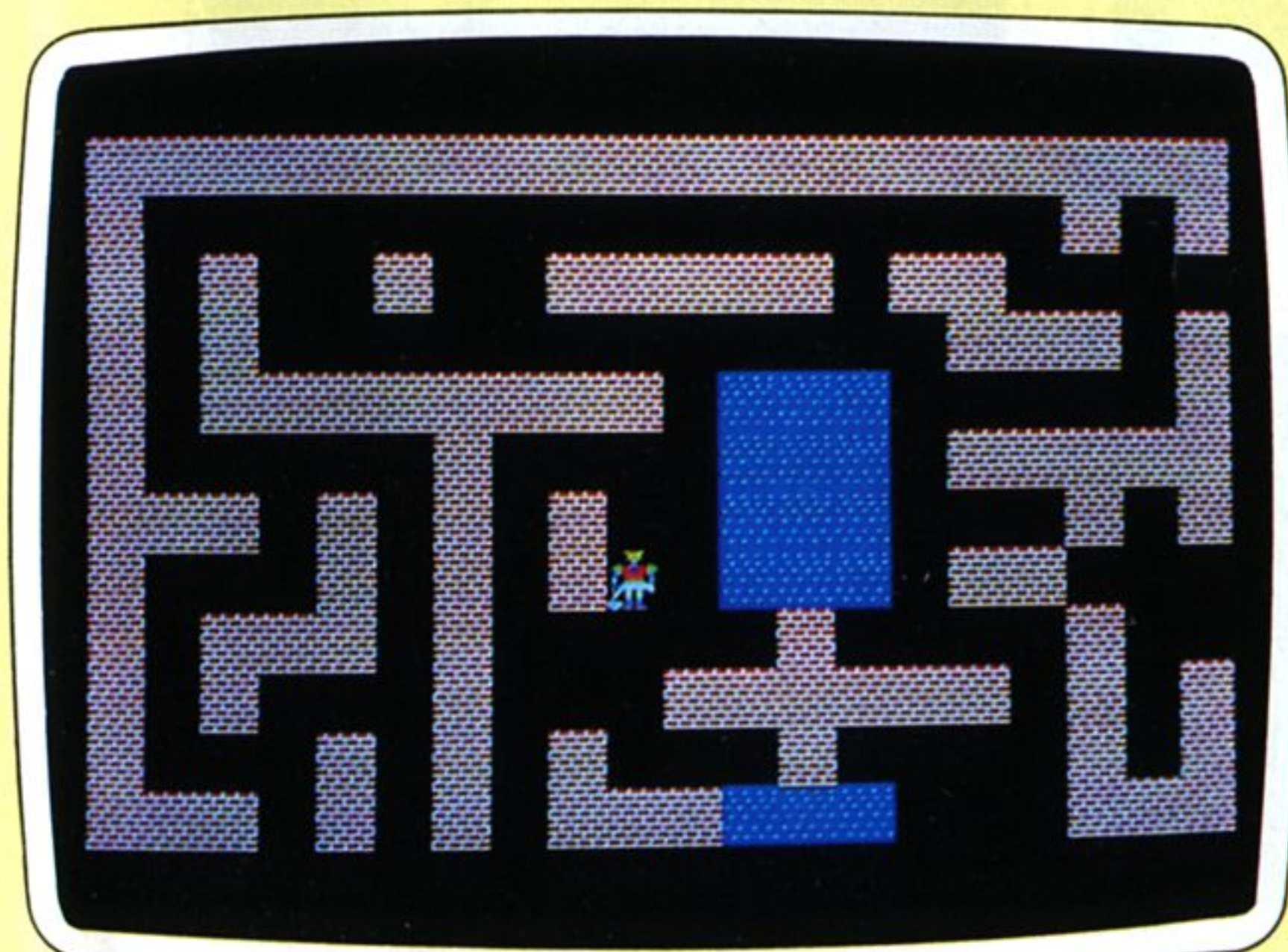
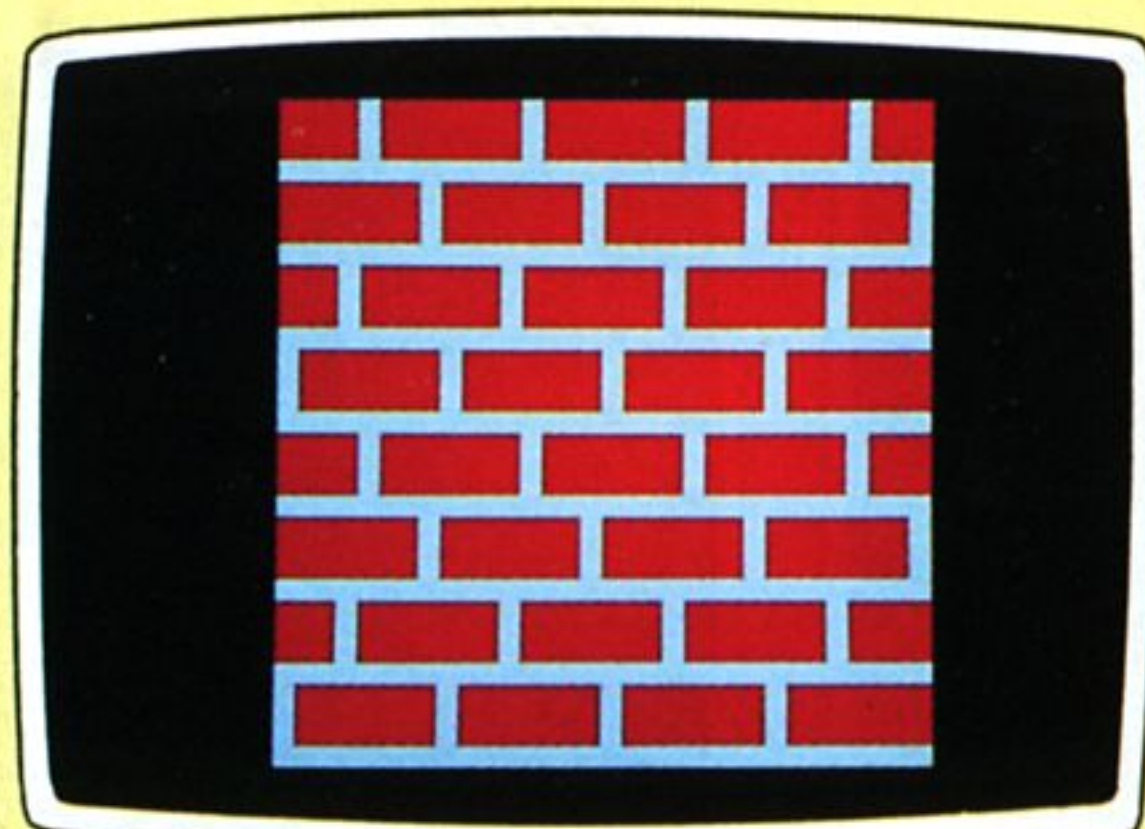
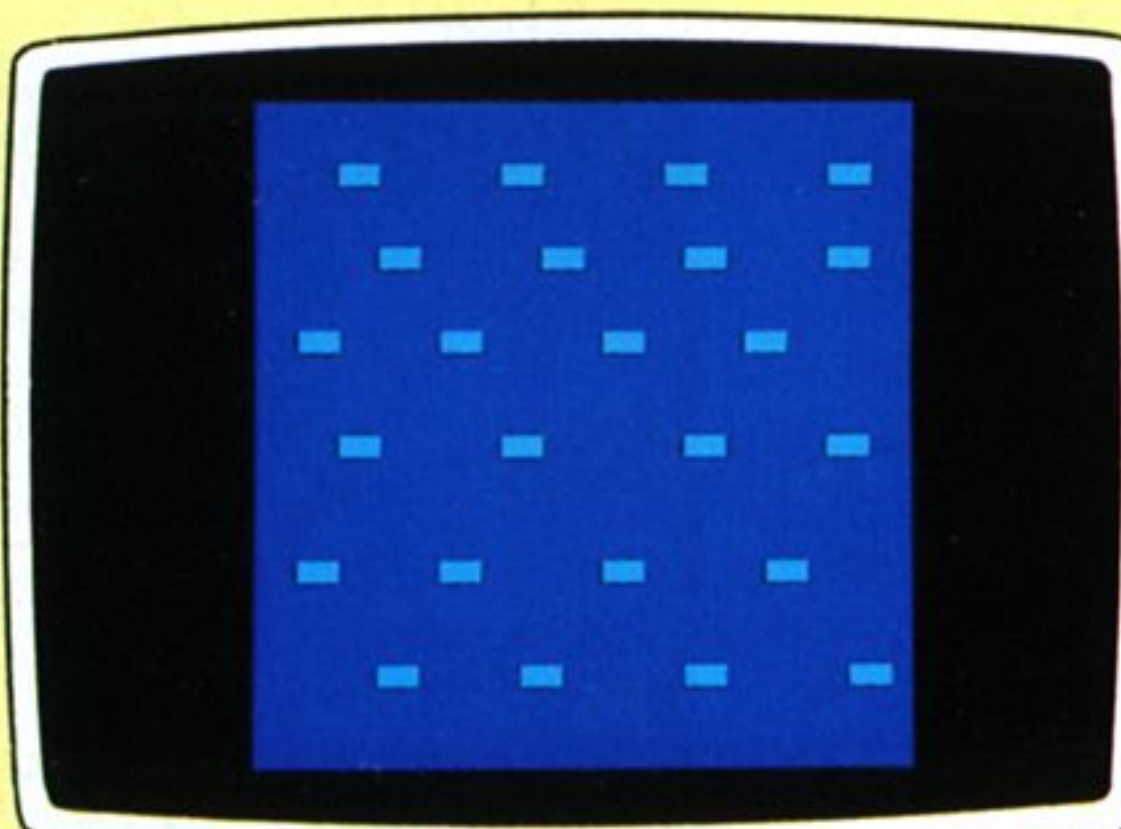
誠文堂新光社

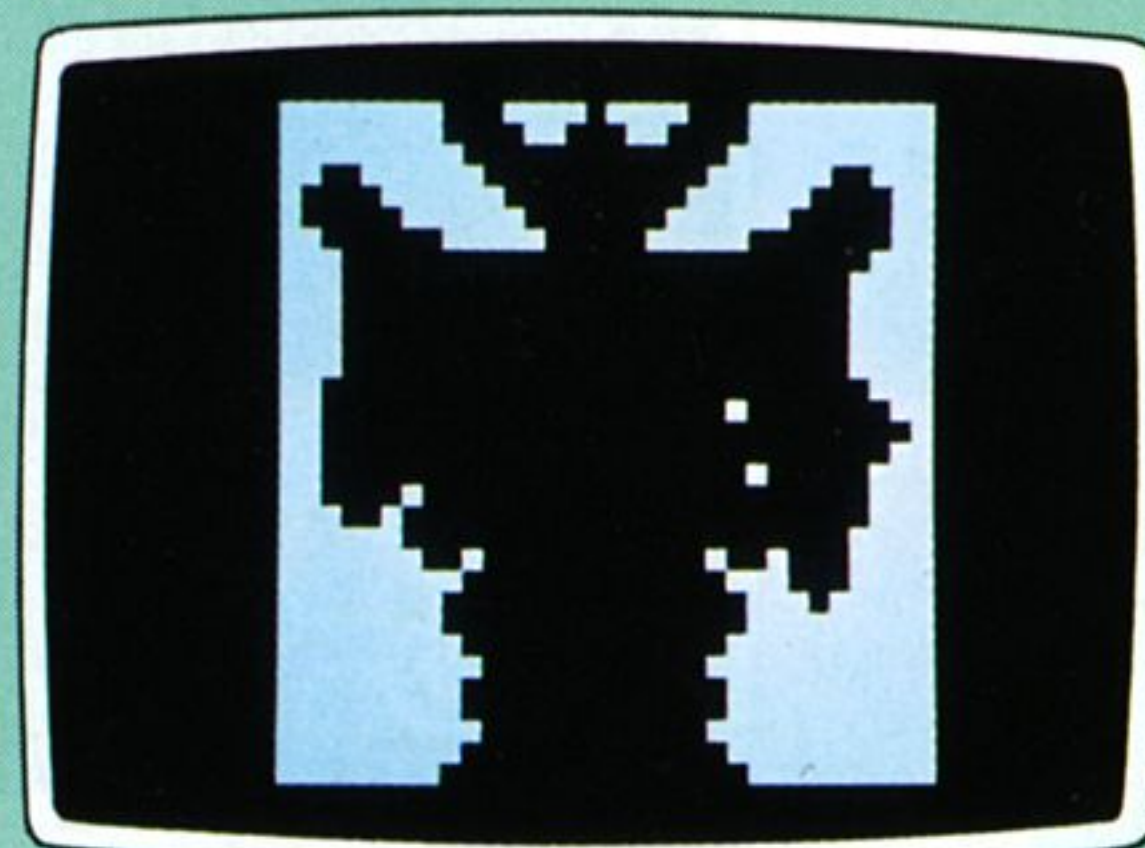
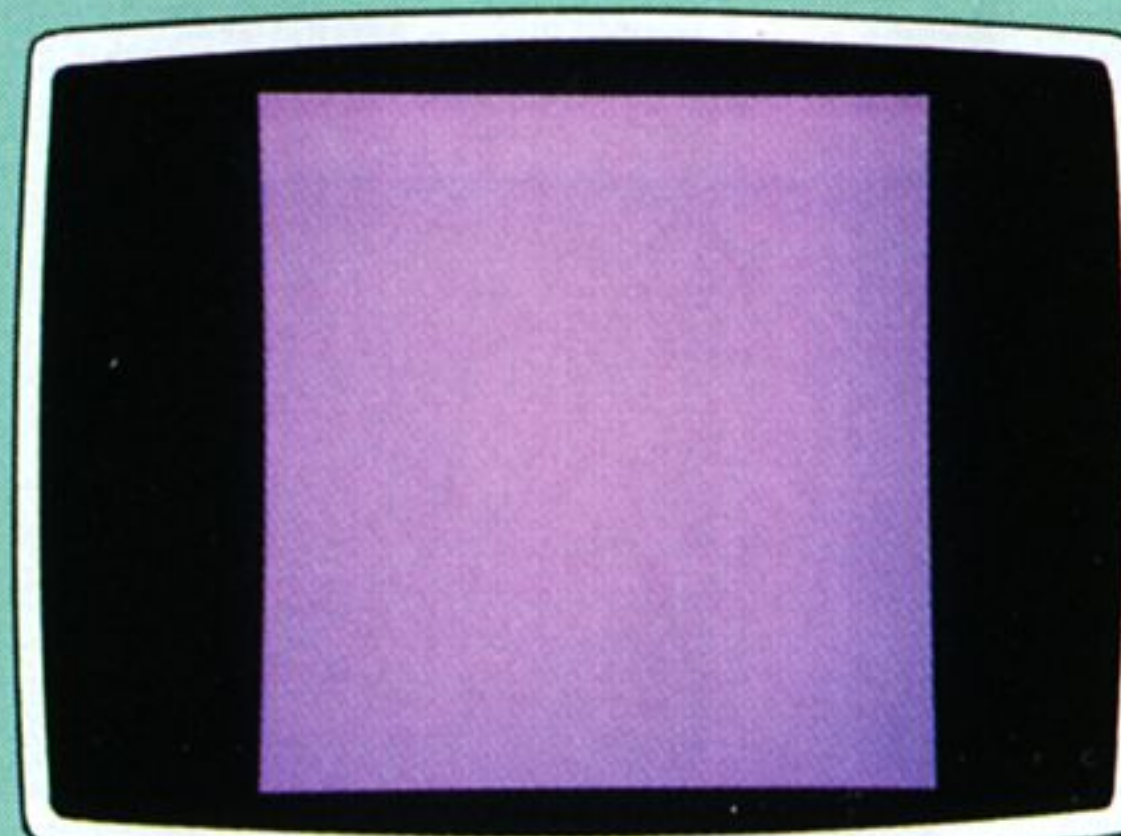
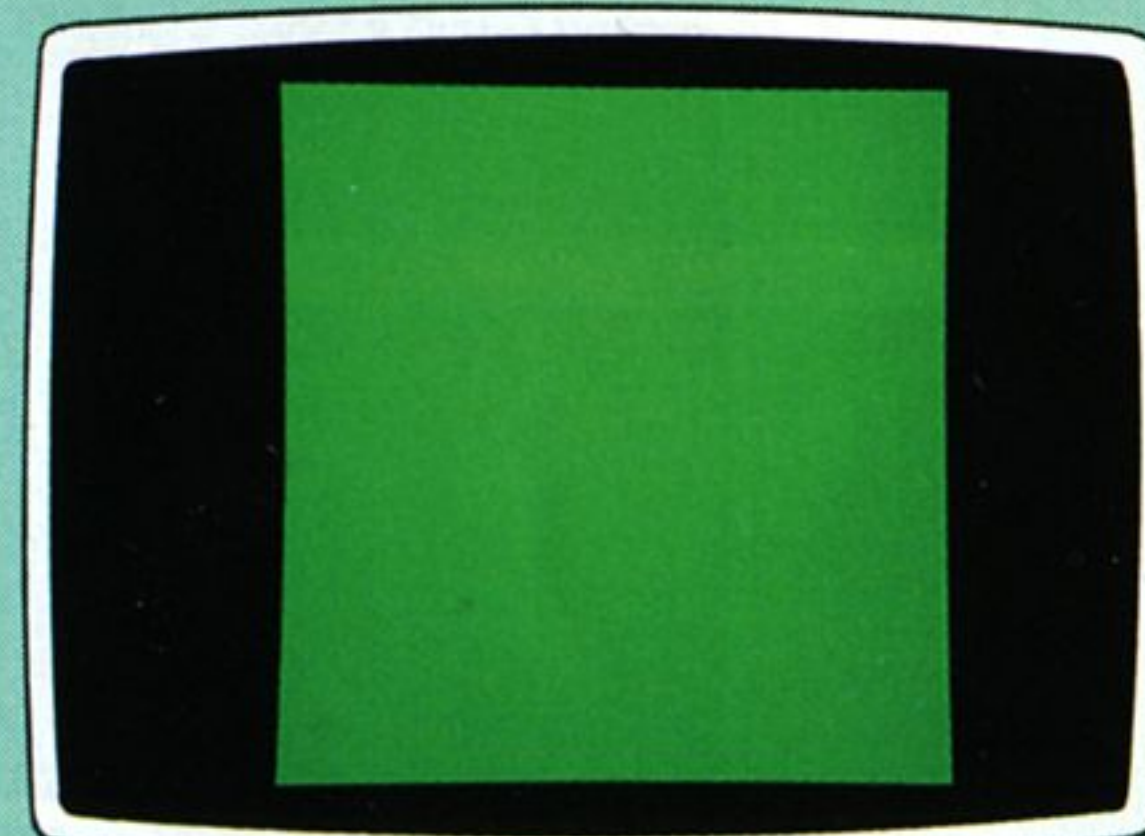
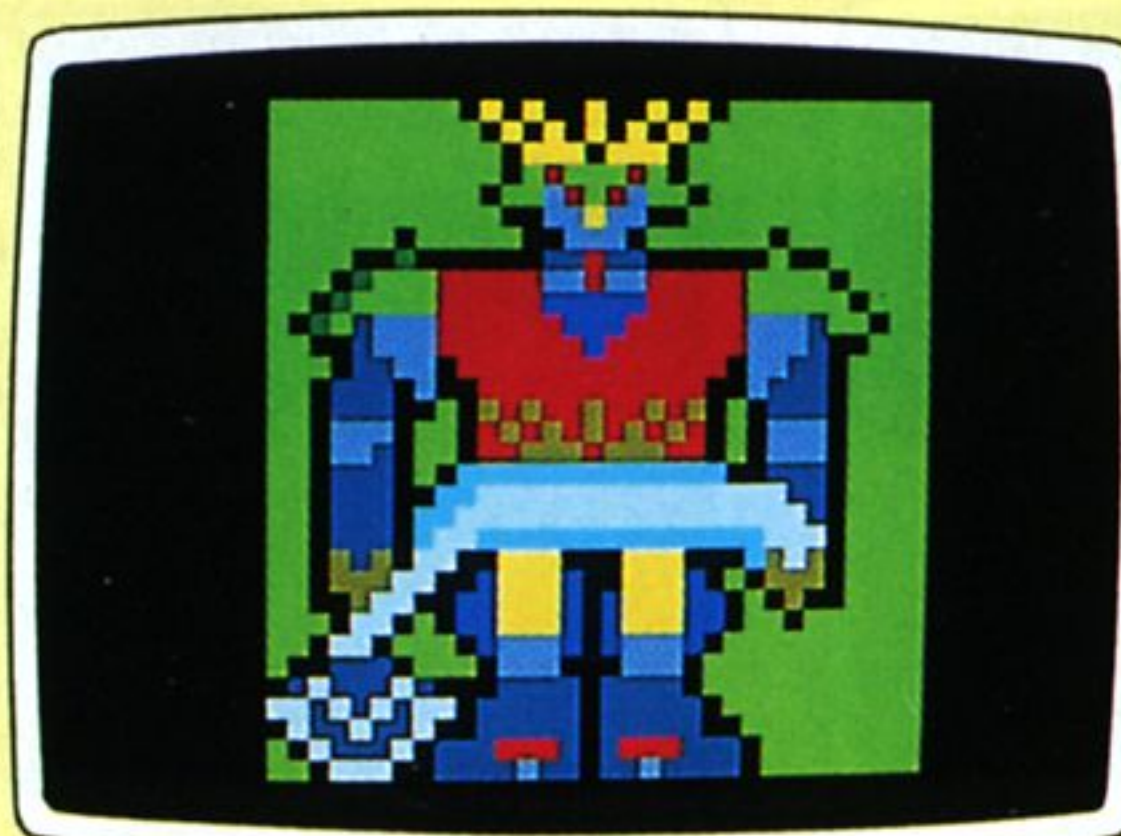
本書に掲載のゲーム画面例

☆キャラクタジェネレータ

どんなキャラクタでも作れる(p.105)

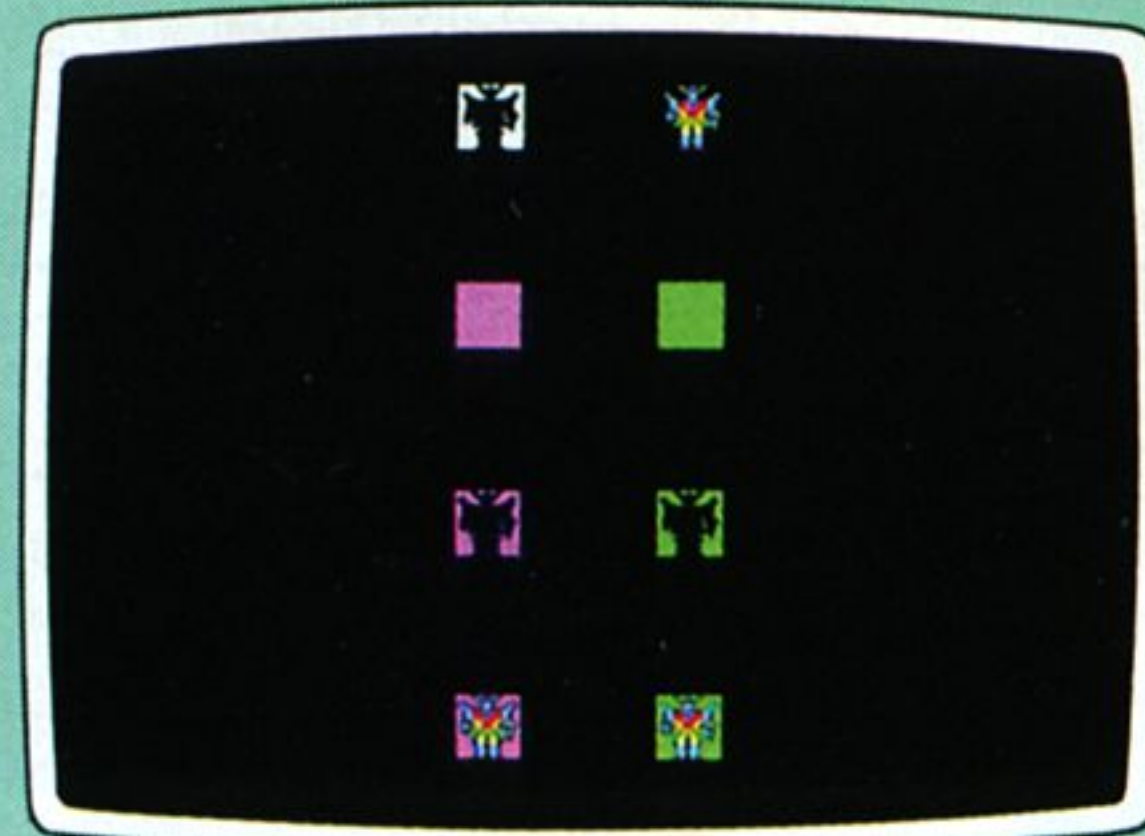
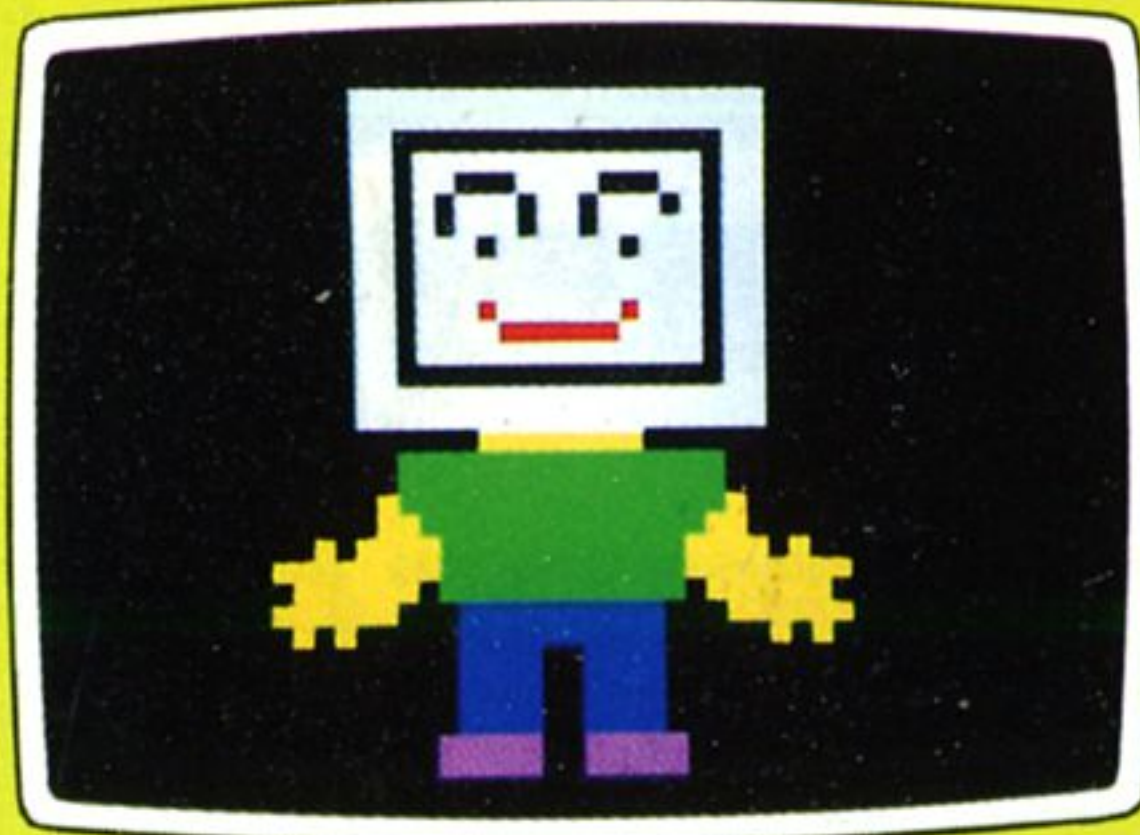
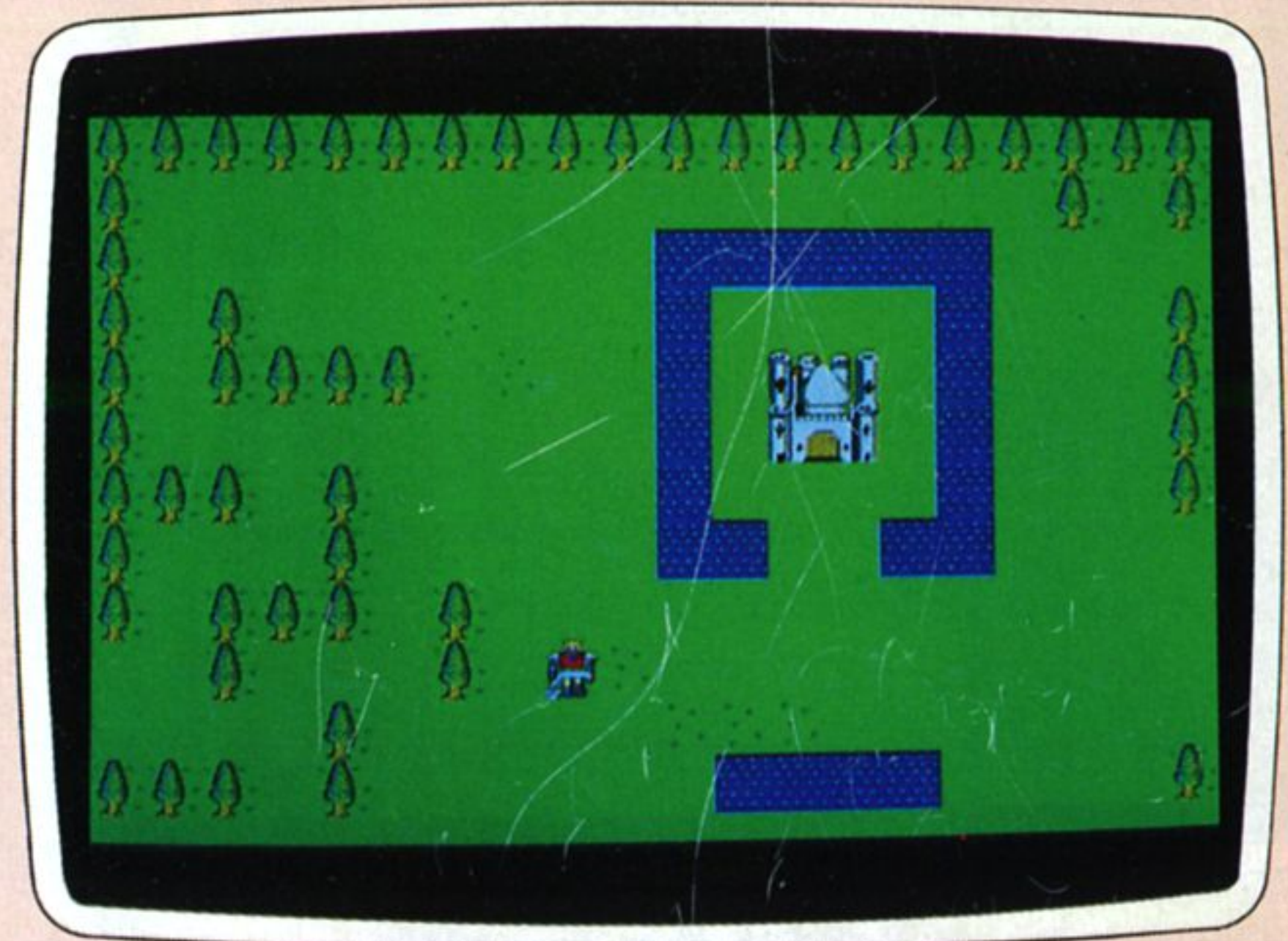
ロボットを作る(p.107)

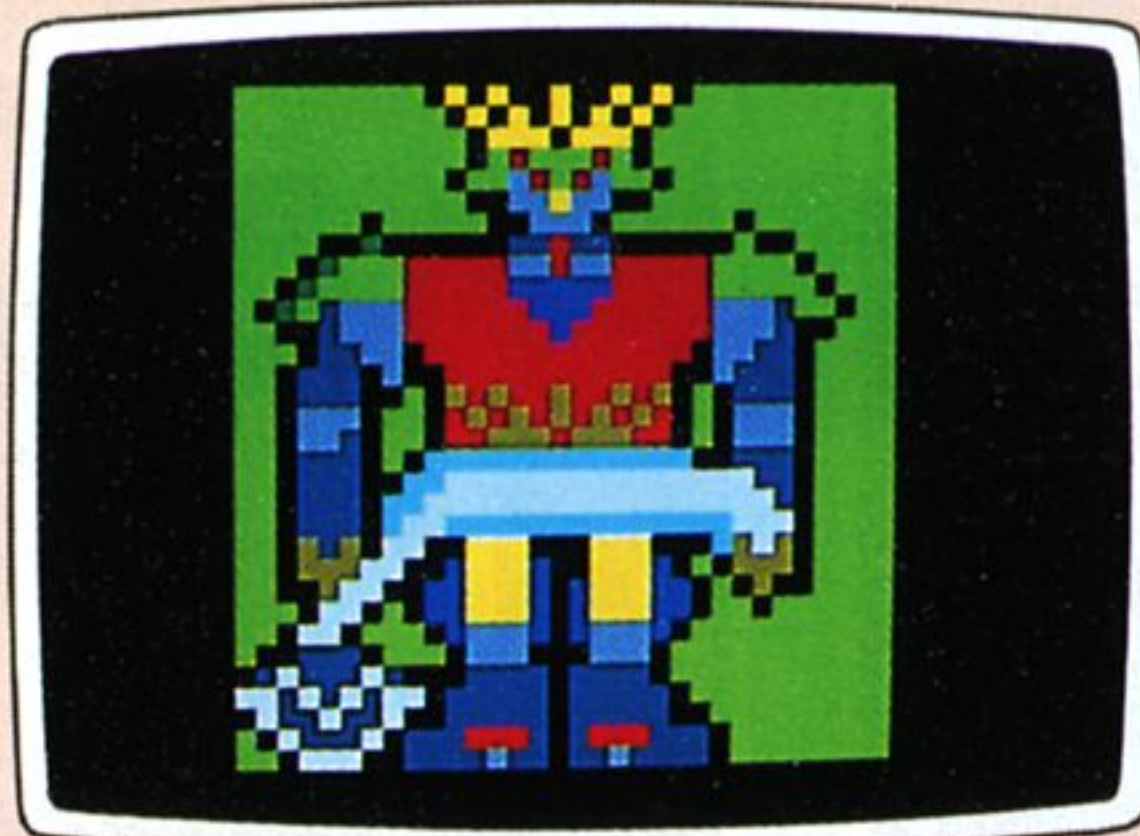
☆迷路ゲーム『MAZE』

『MAZE』迷路ゲーム(p.126)

☆重ね合わせ

DELICE(p.122)

透明(p.122)

全面ピンク(p.122)

全面緑(p.122)

EXDELIC(p.128)

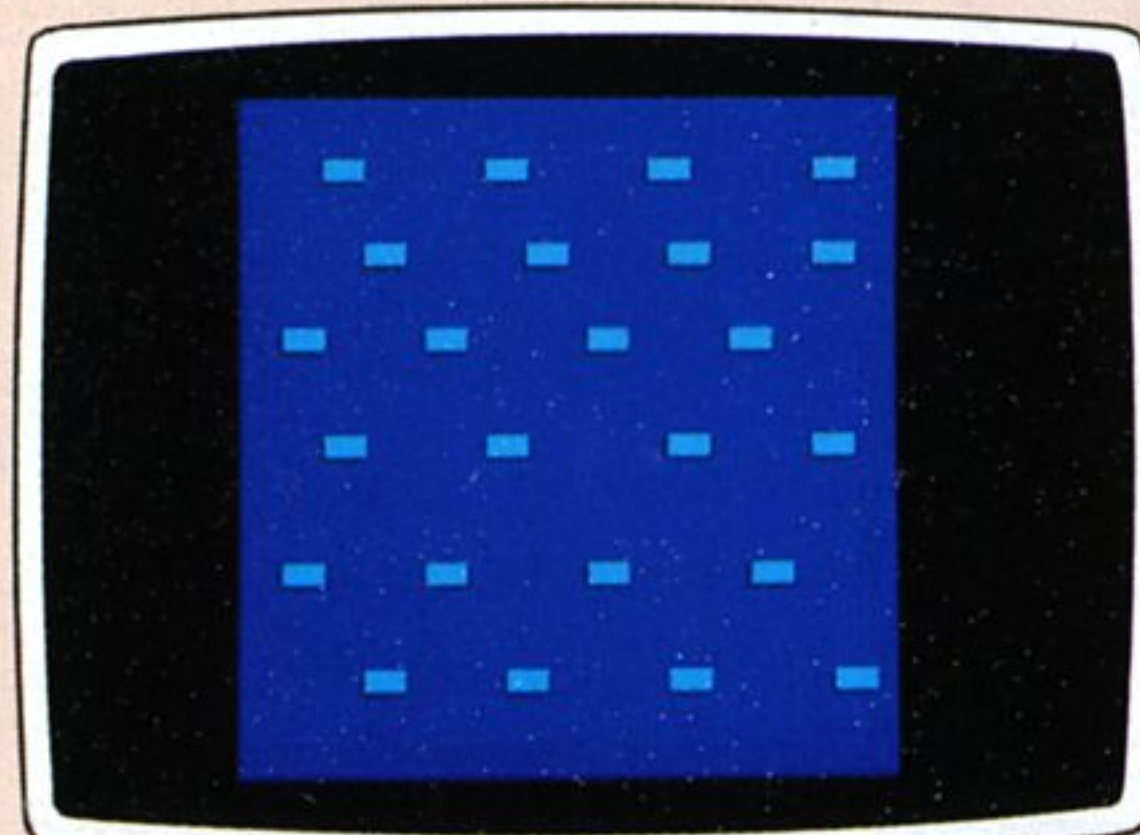
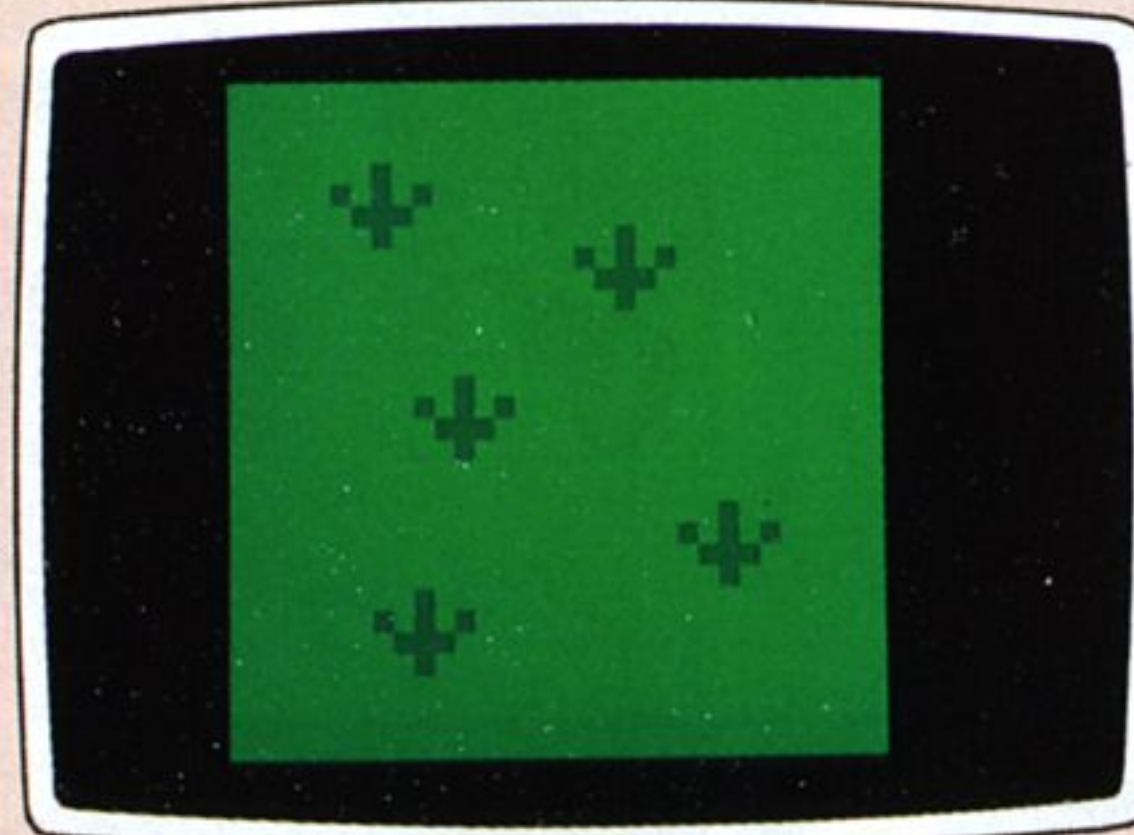
重ね合わせの原理(p.124)

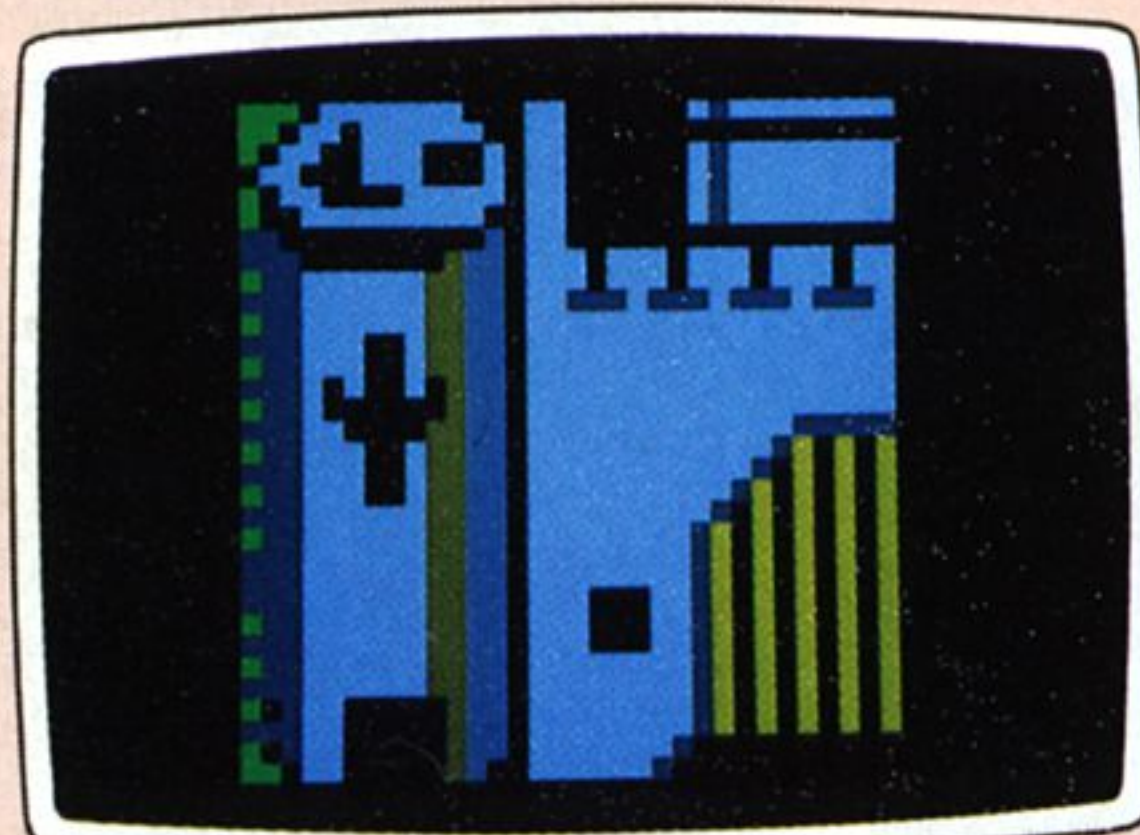
レンガ壁(p.128)

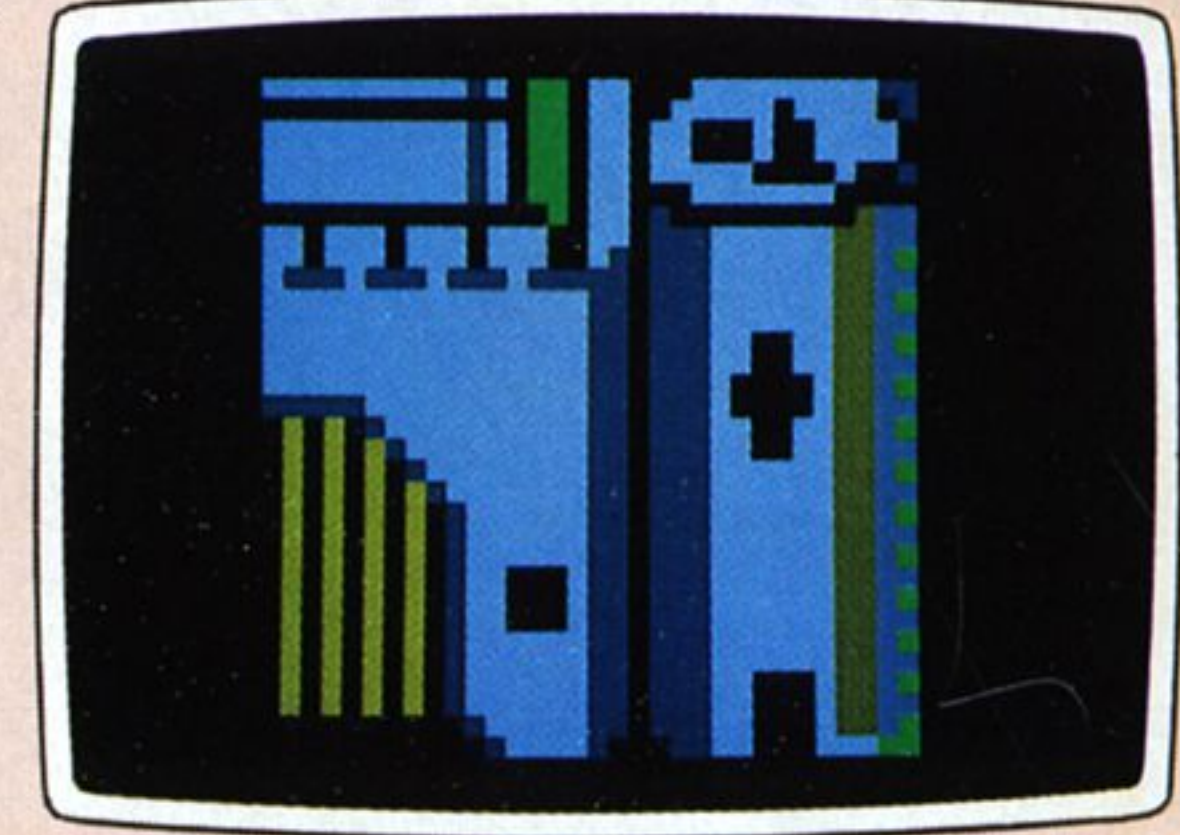
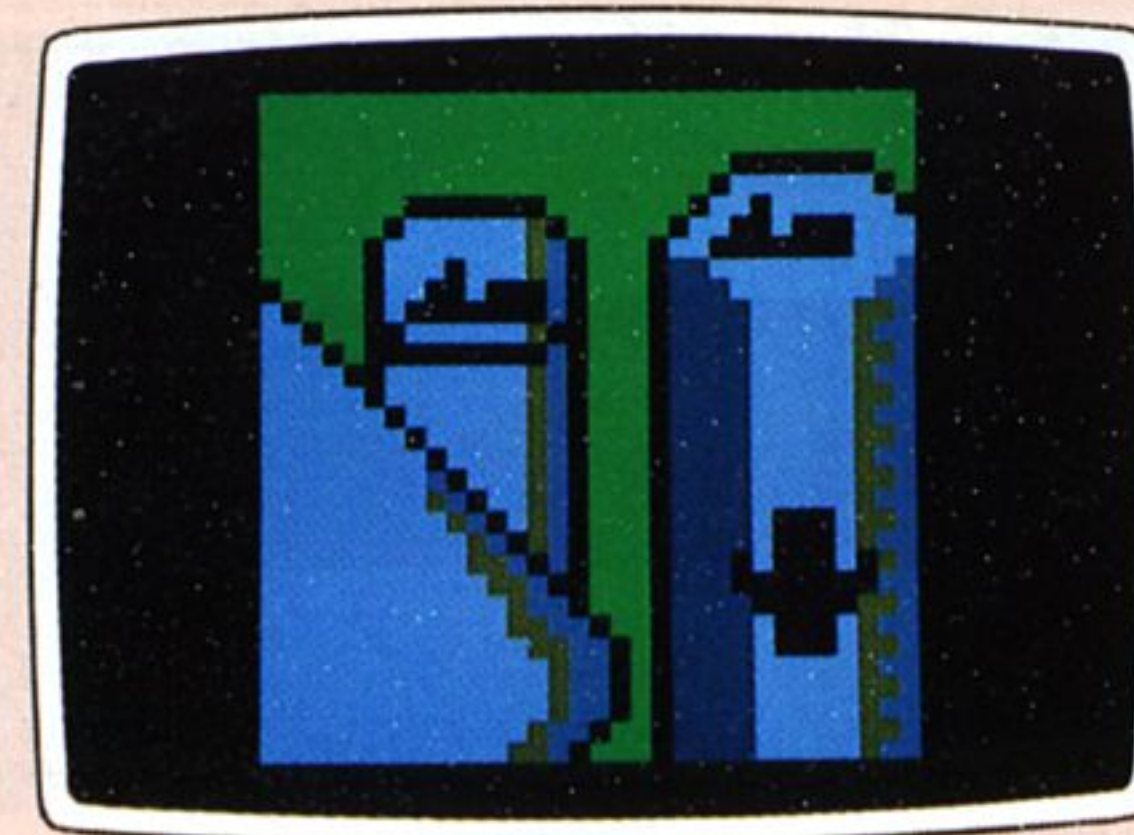
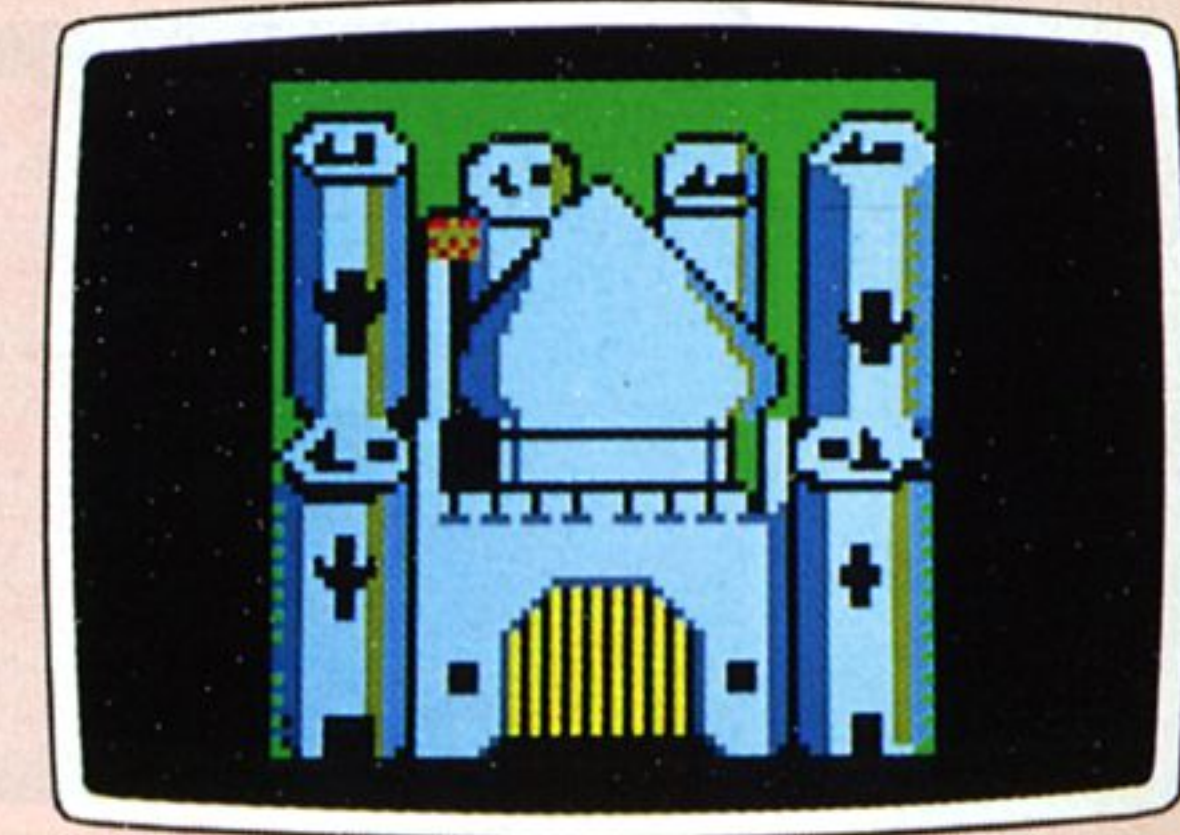
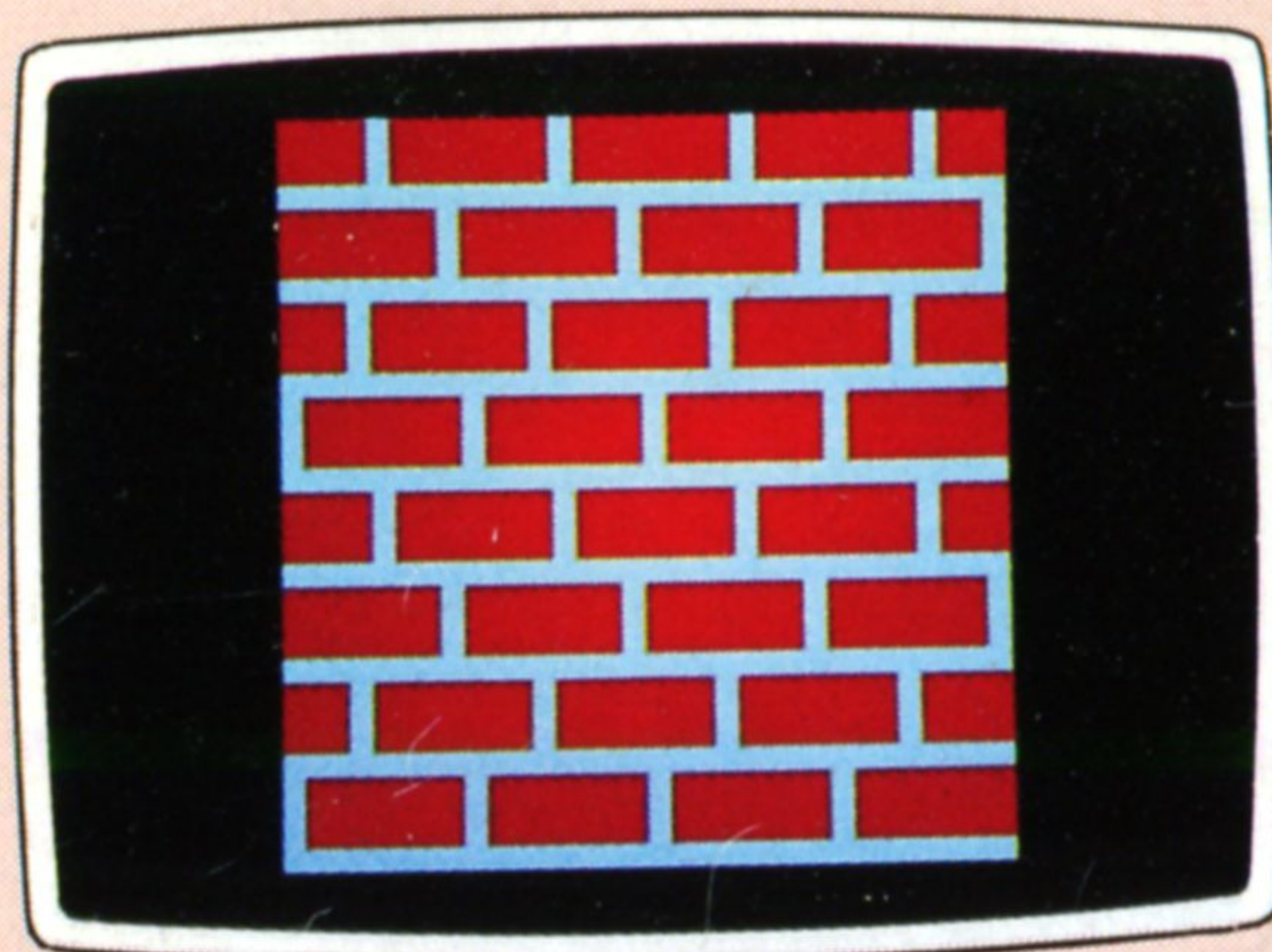
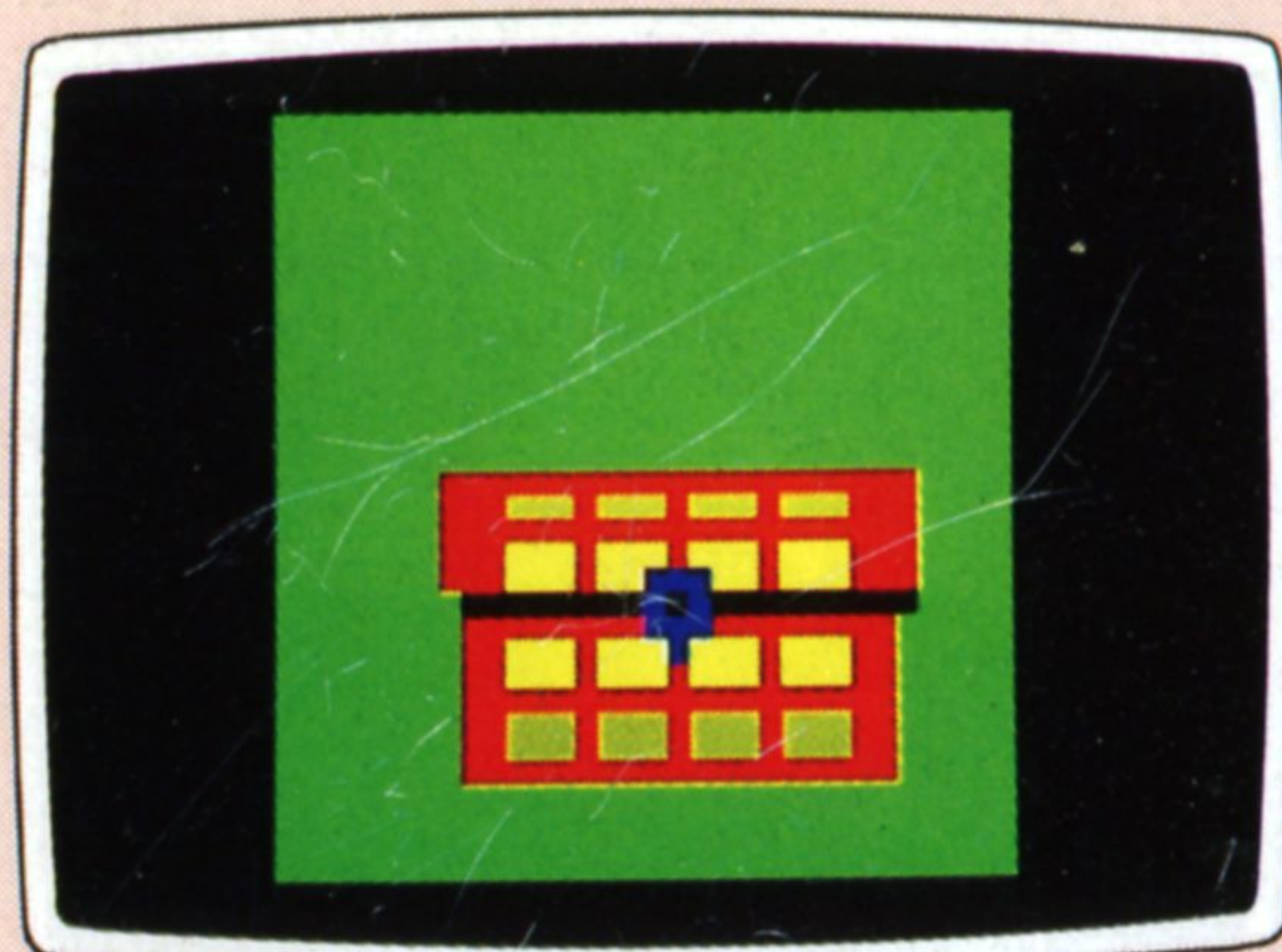
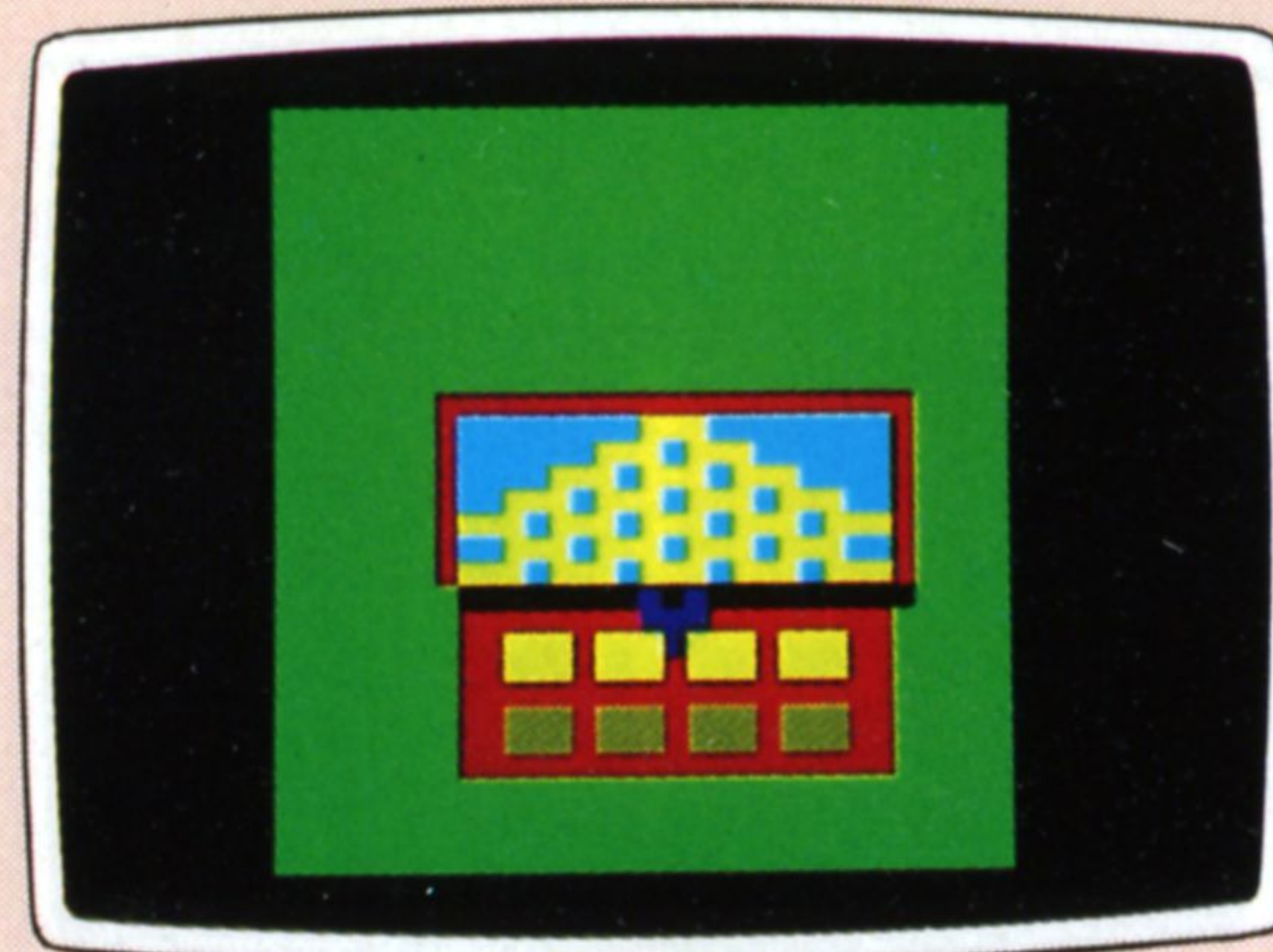
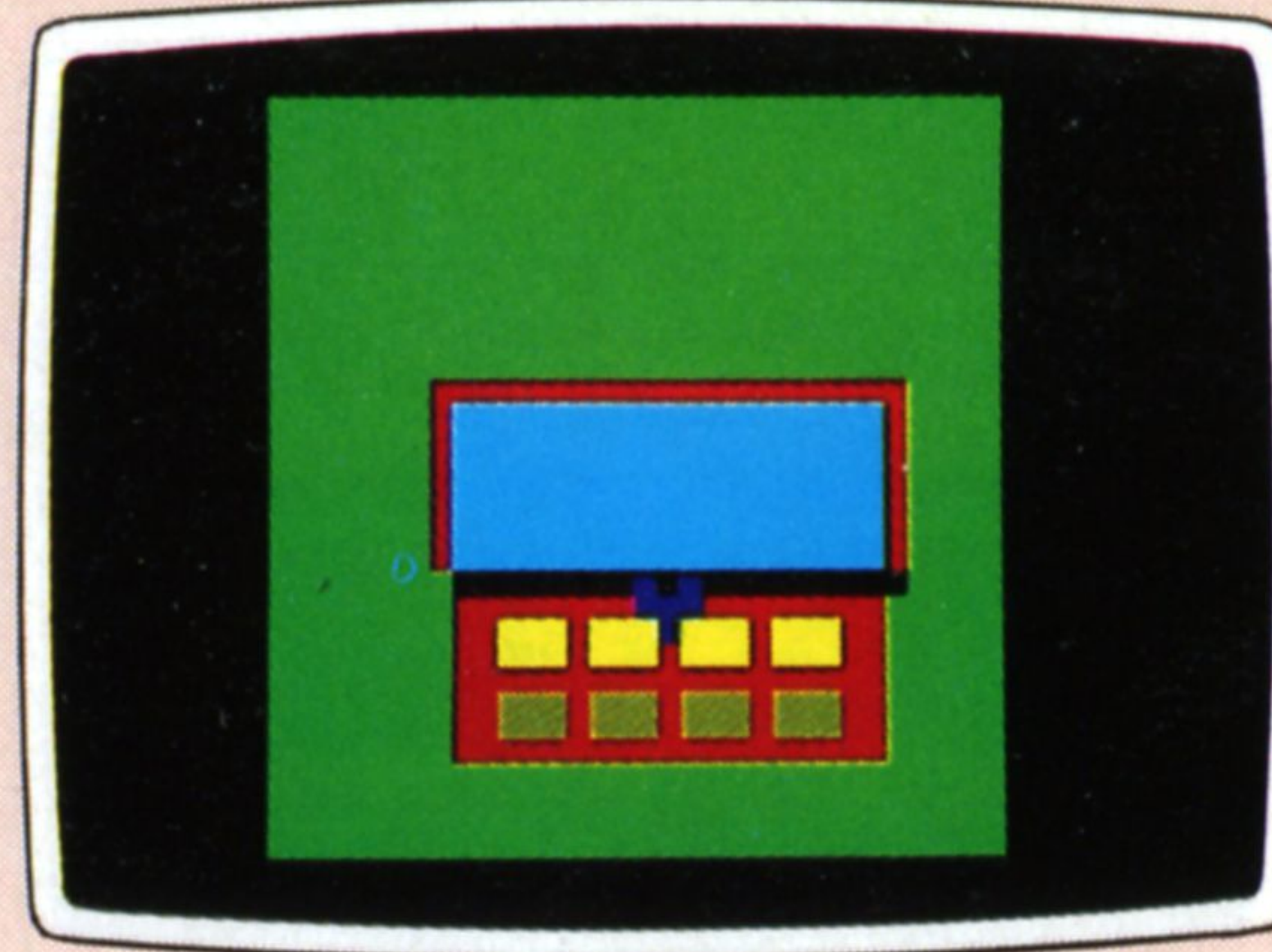
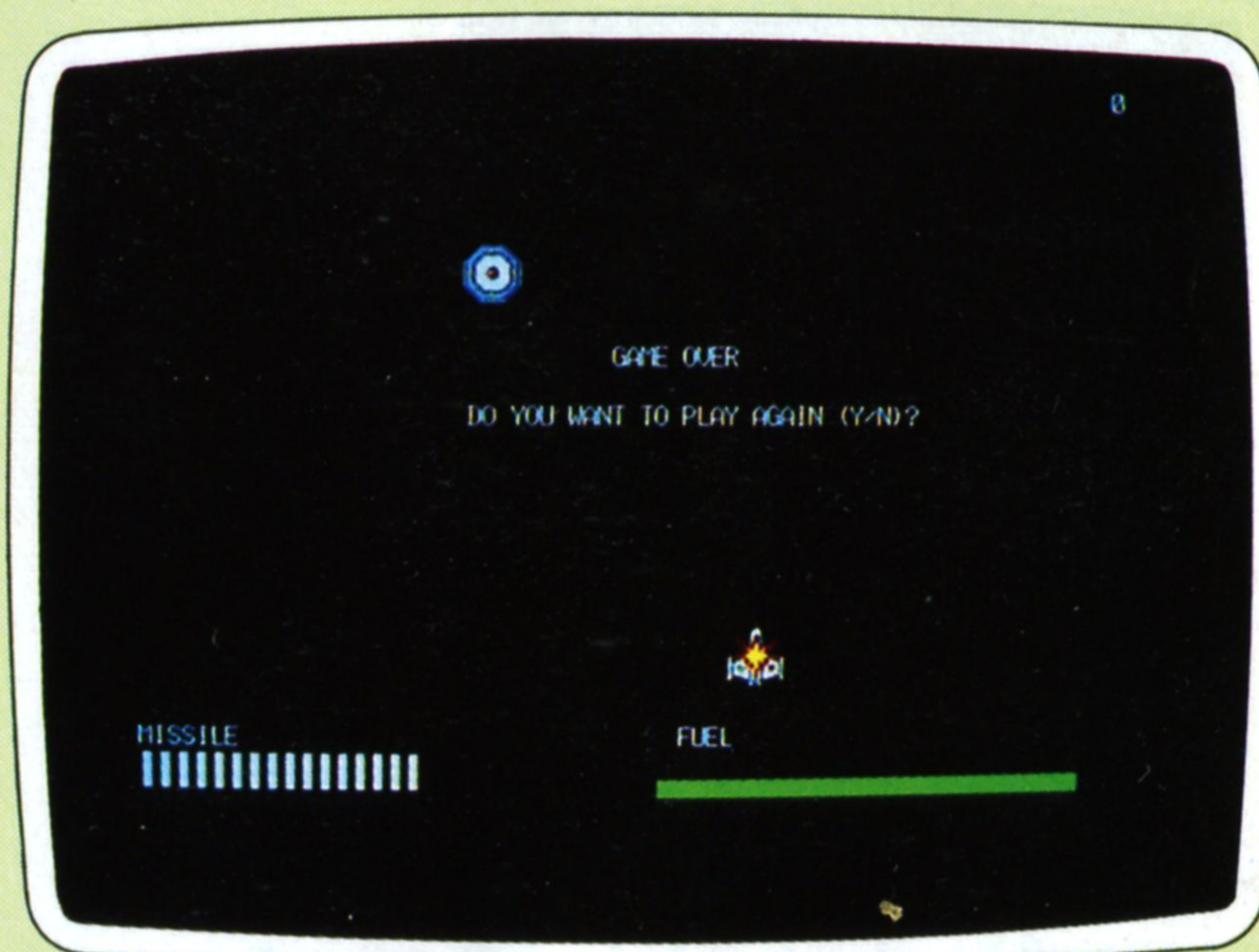
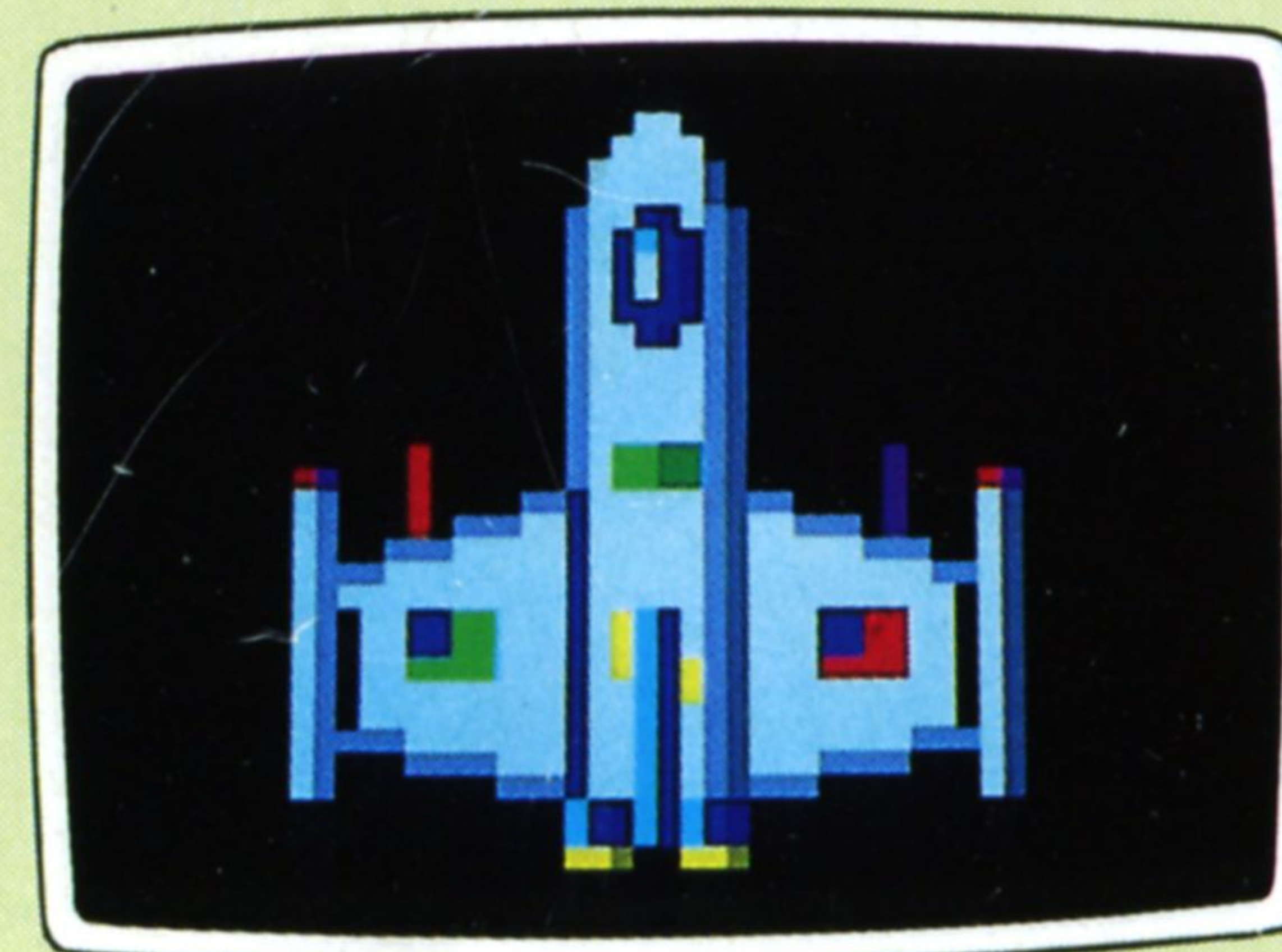
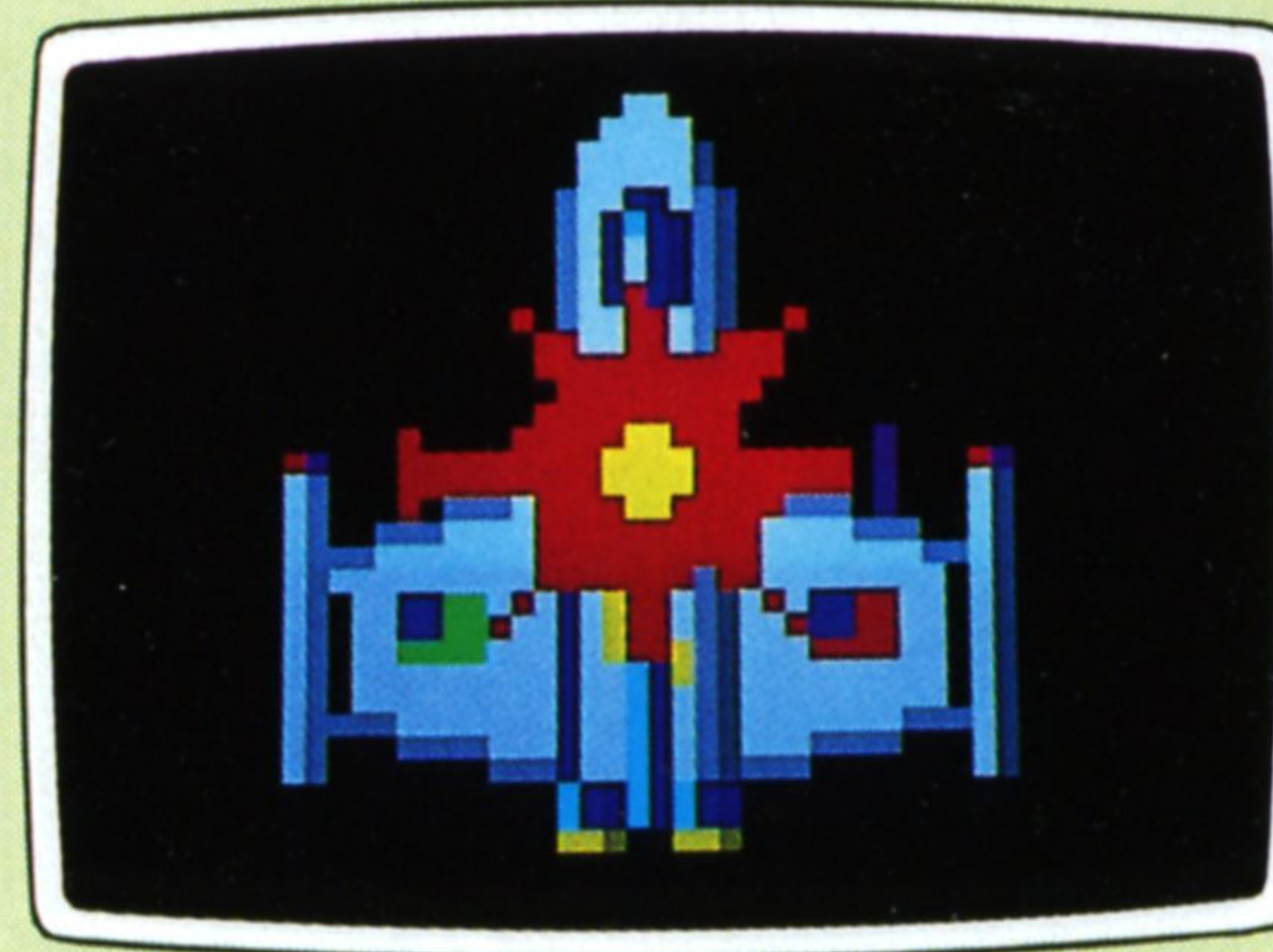
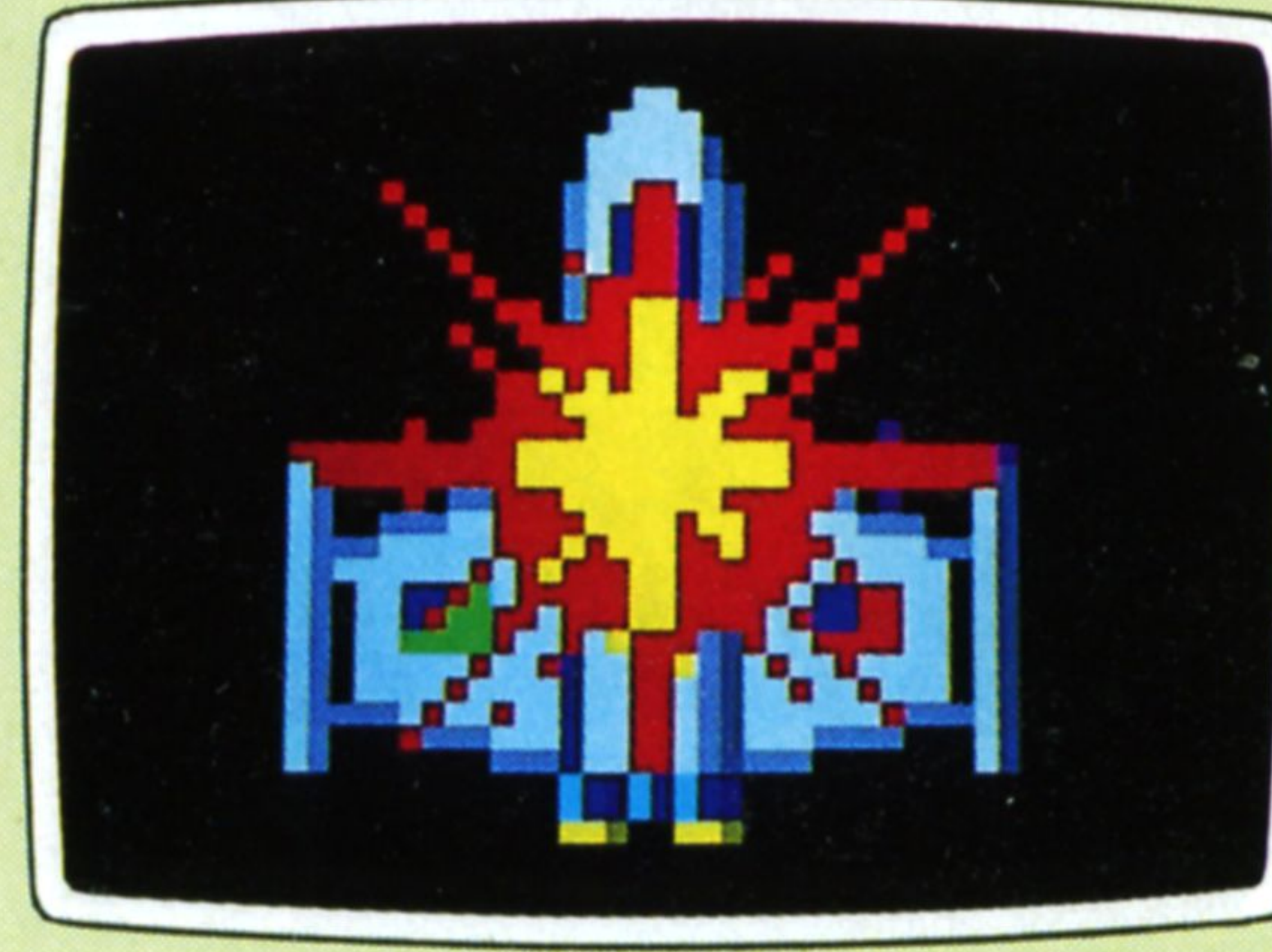
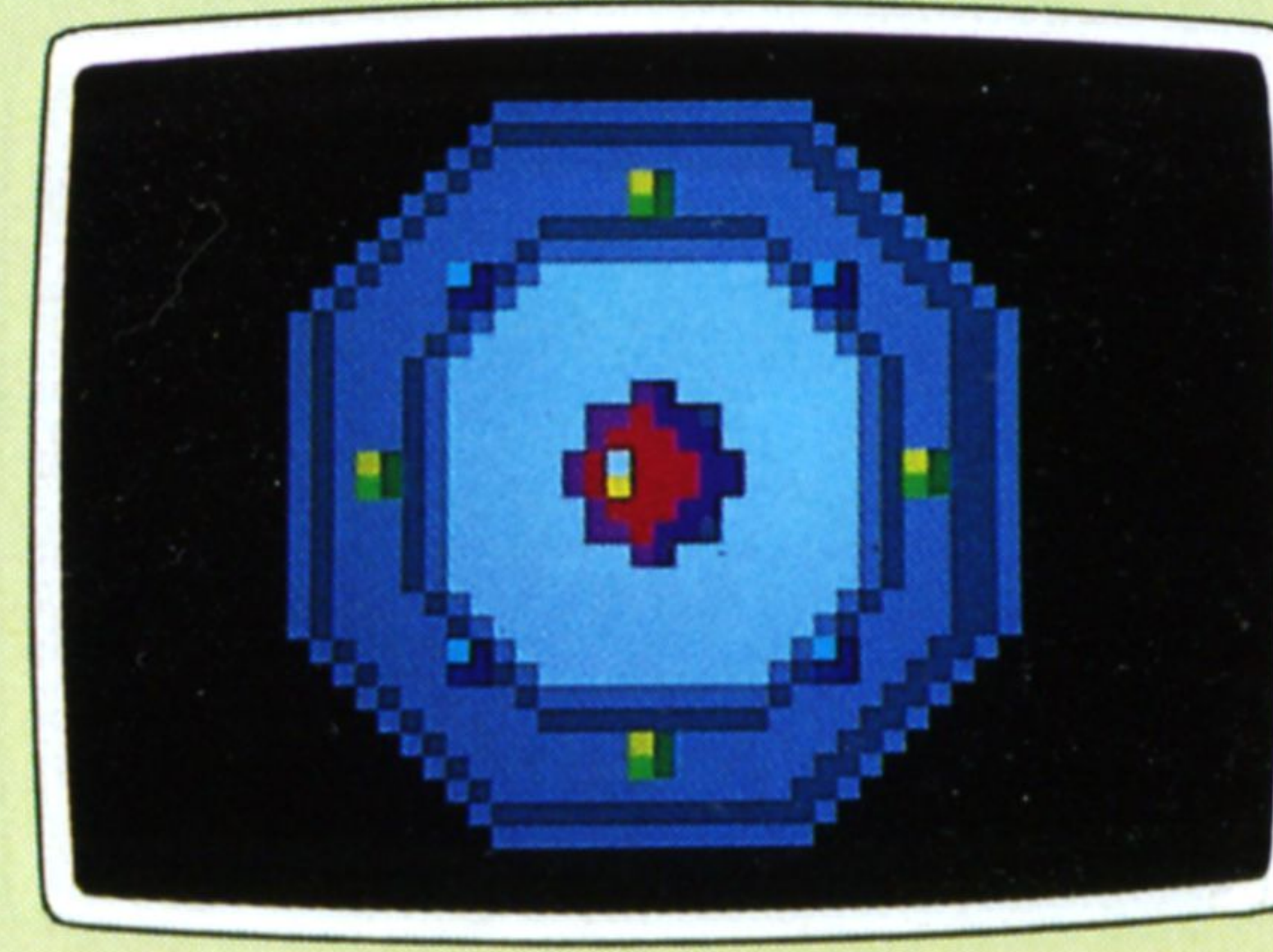
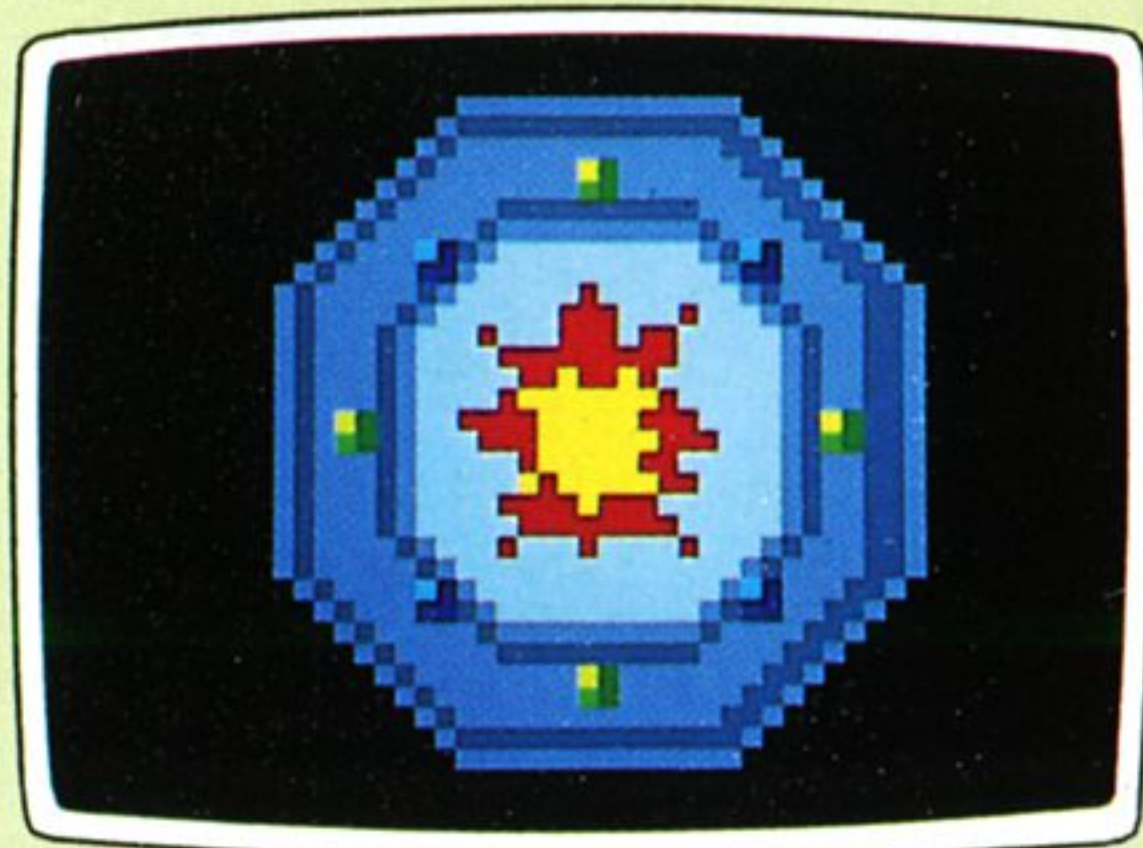
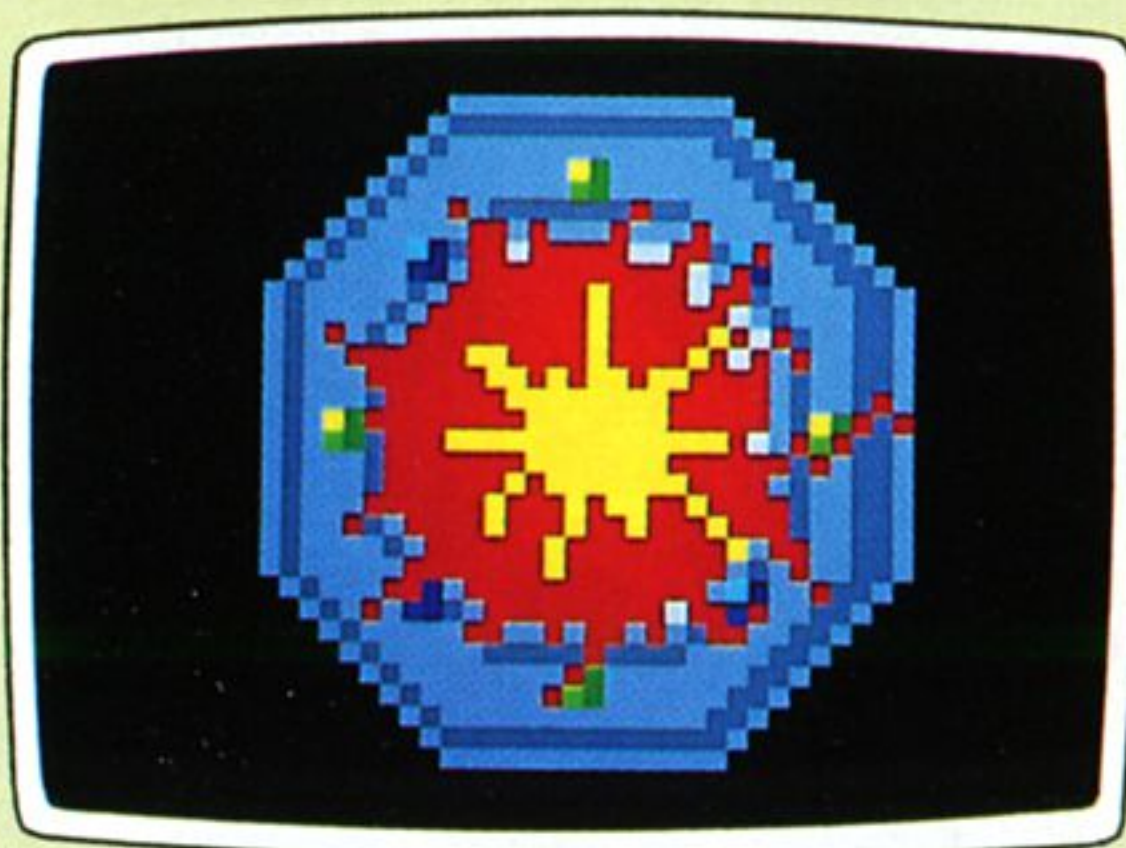
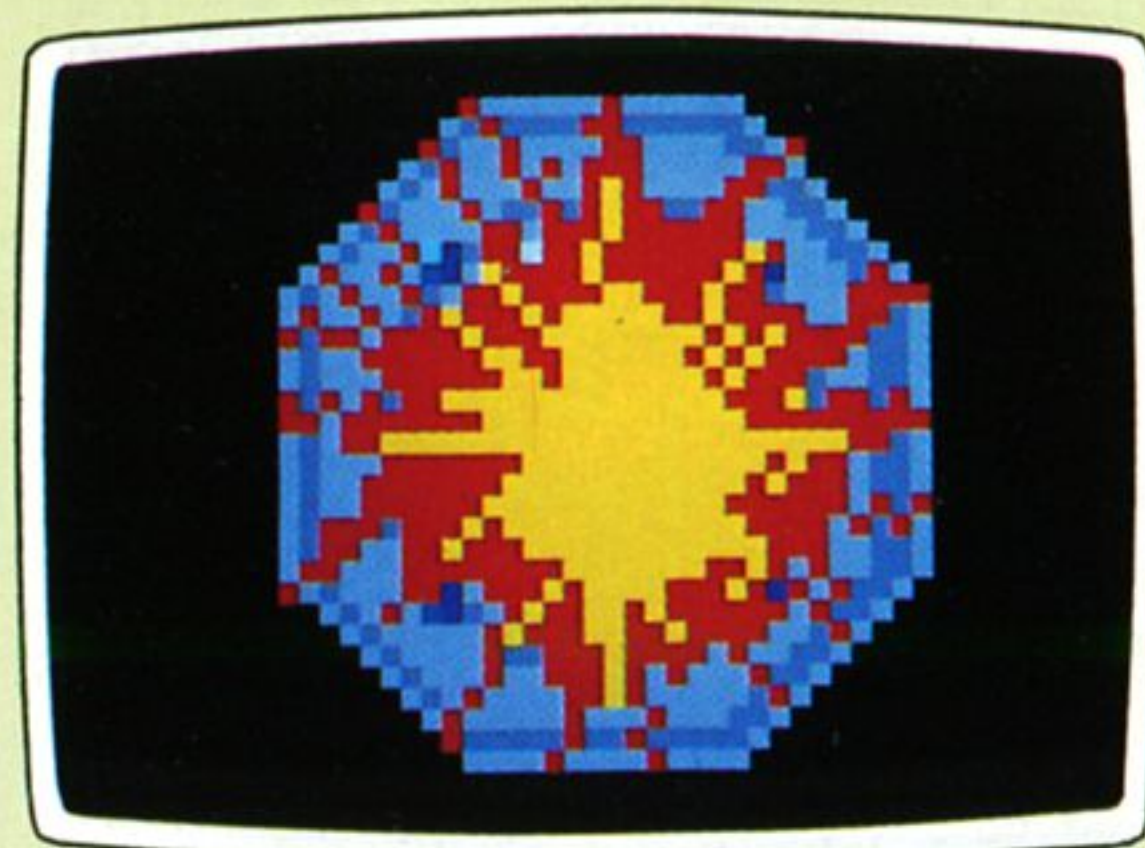
水面(p.128)

(注)各ネームのラストの(　)中は、本文に掲載されているモノクロ写真ページです。

『My name is HERO』(p.114)

☆『デライス対ゲドバ』

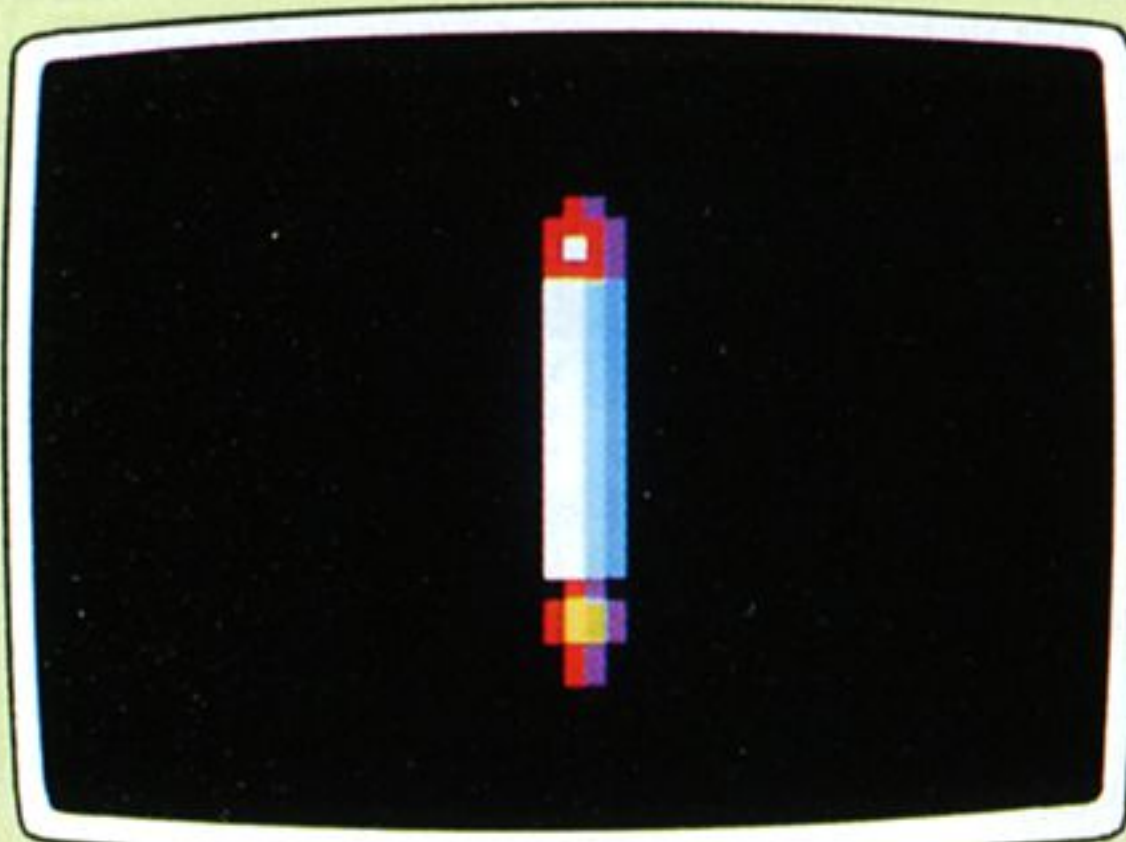
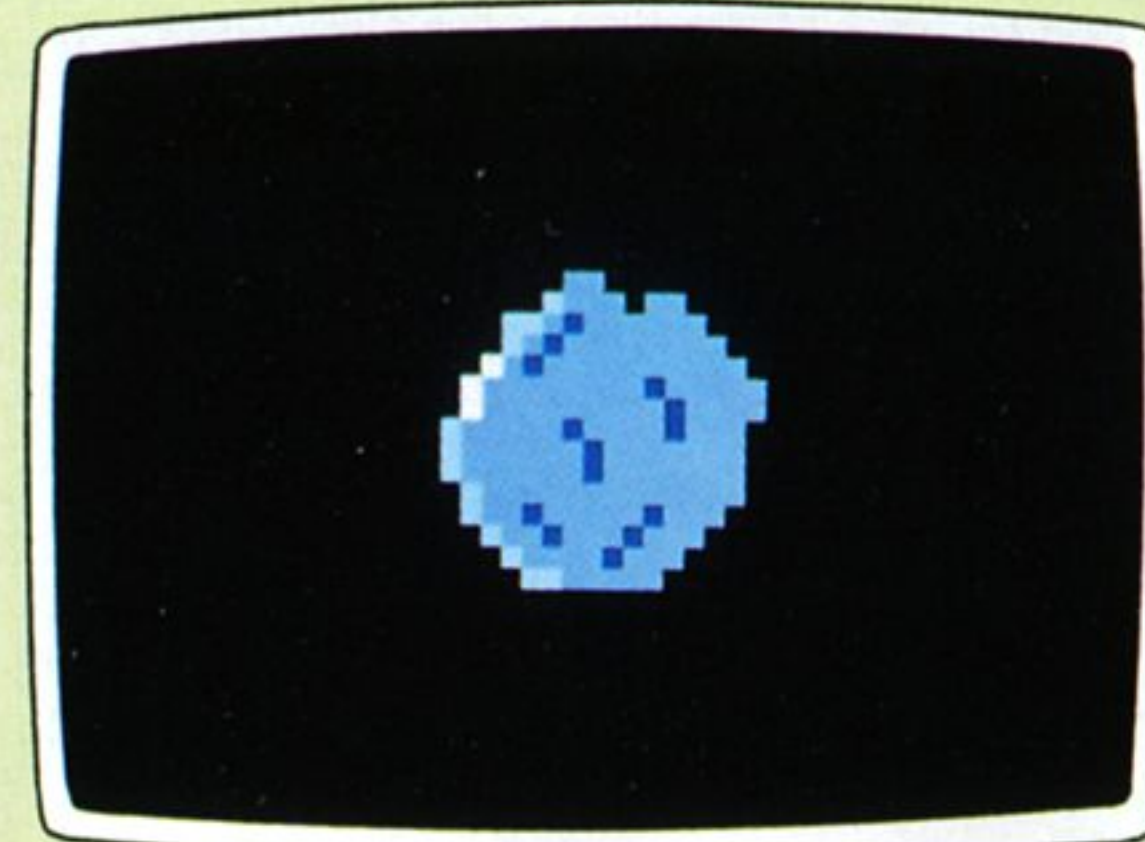
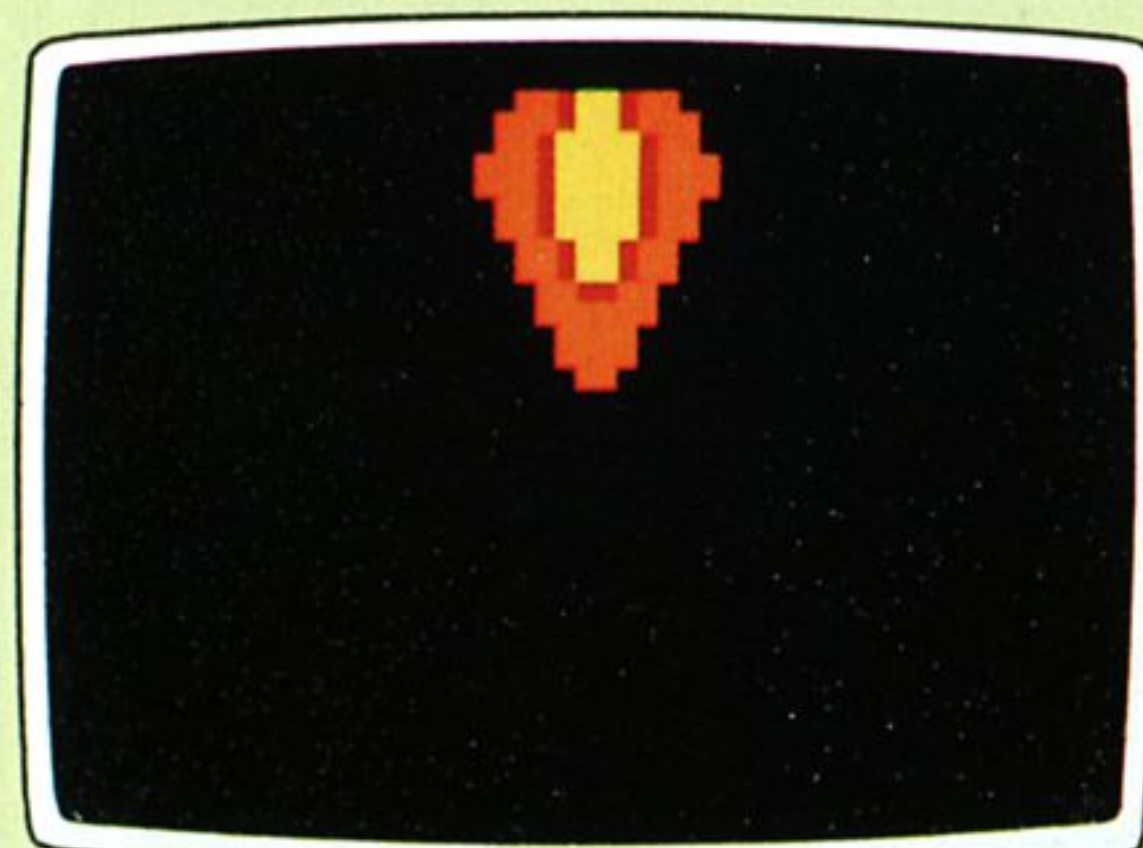
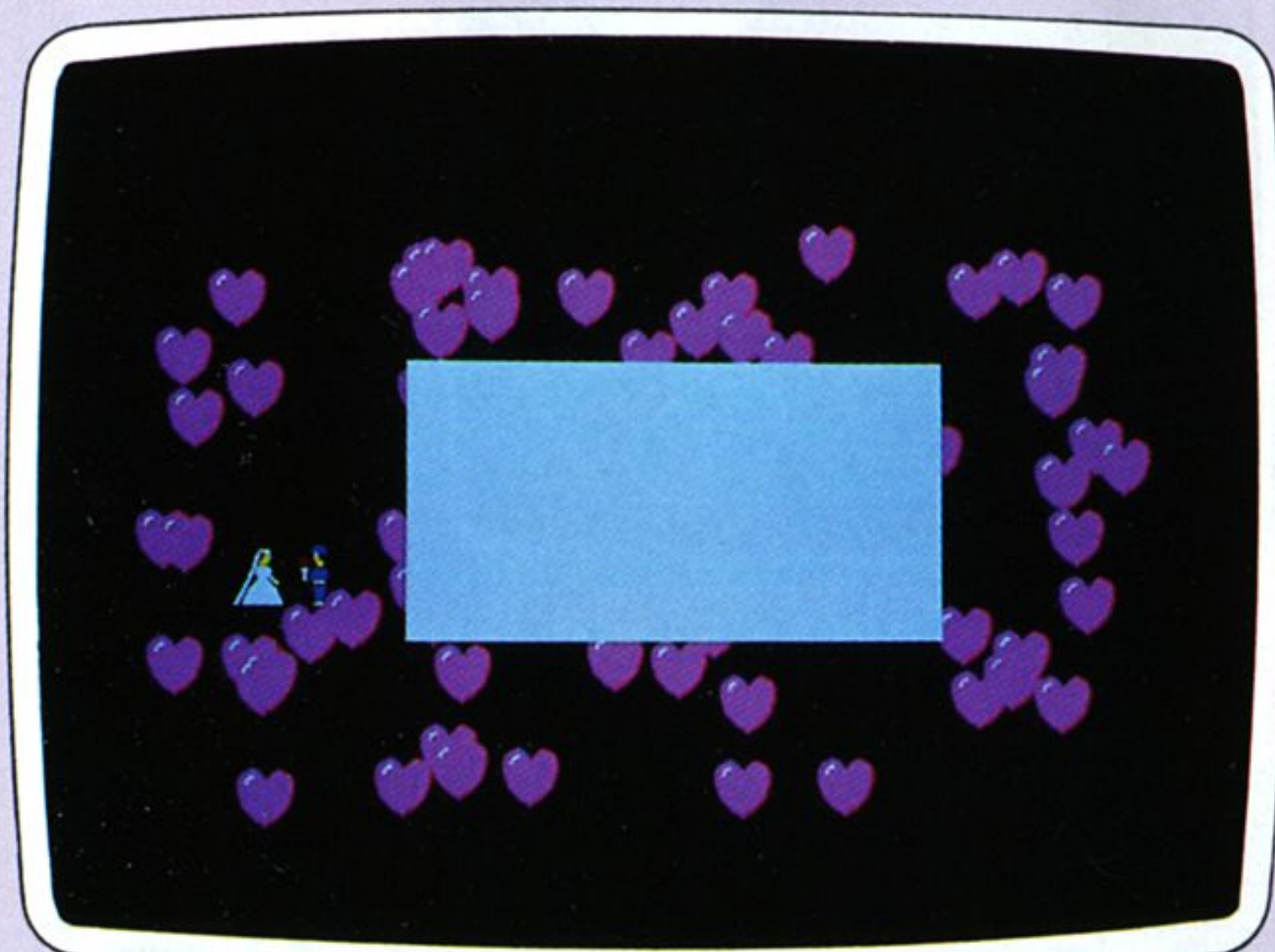
『デライス対ゲドバ』ロールプレイングゲーム(p.135)

緑地，デライス直立(p.136)

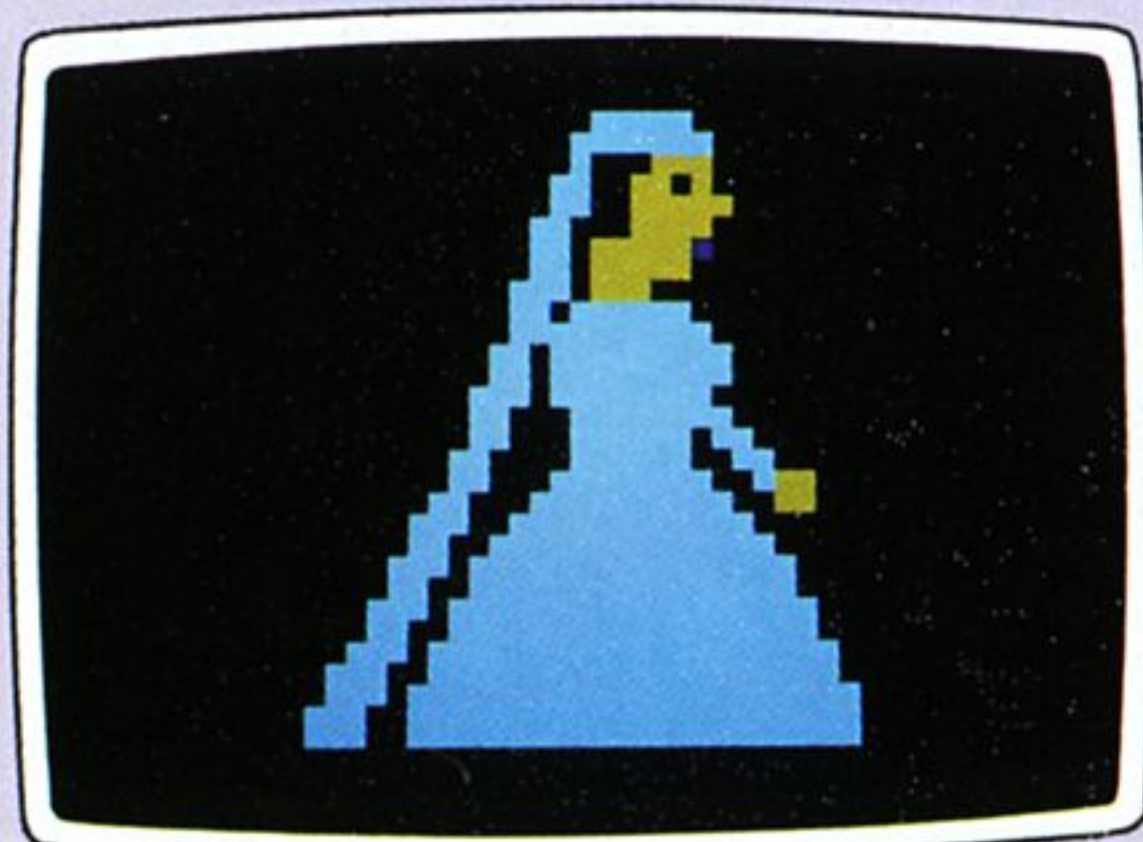
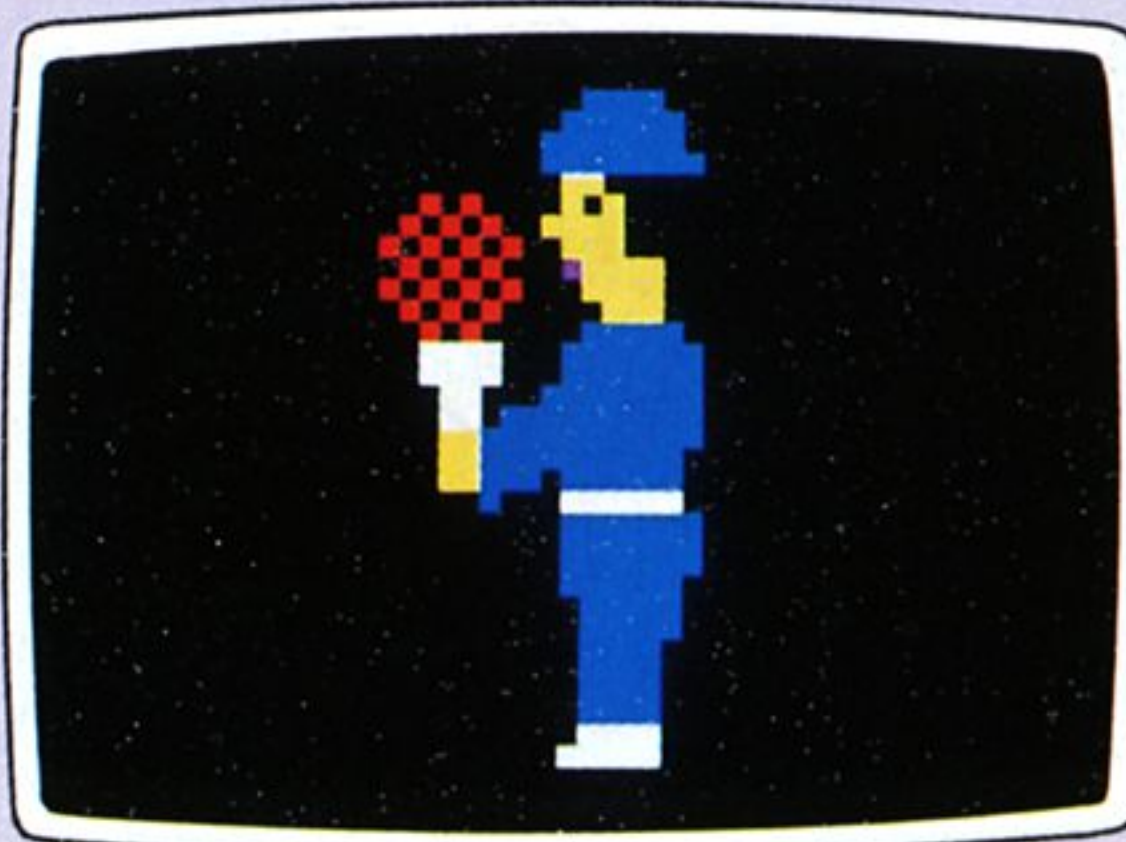
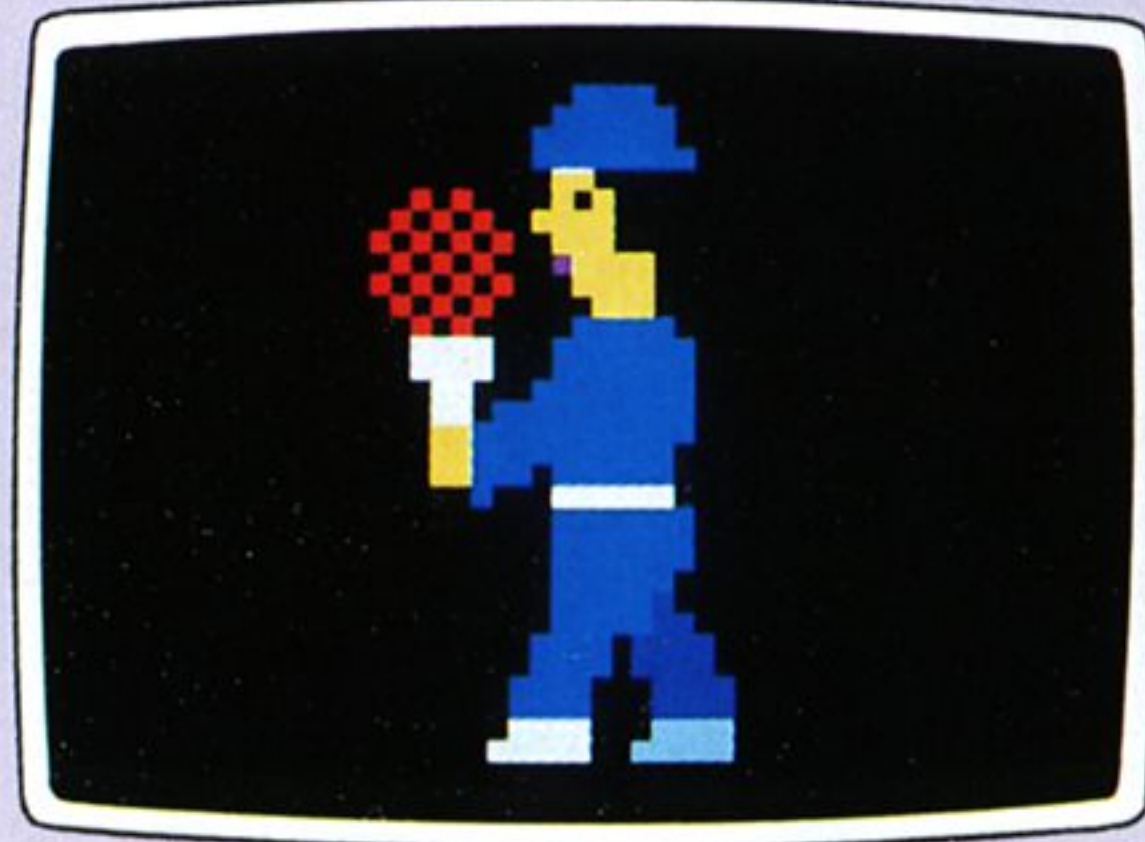
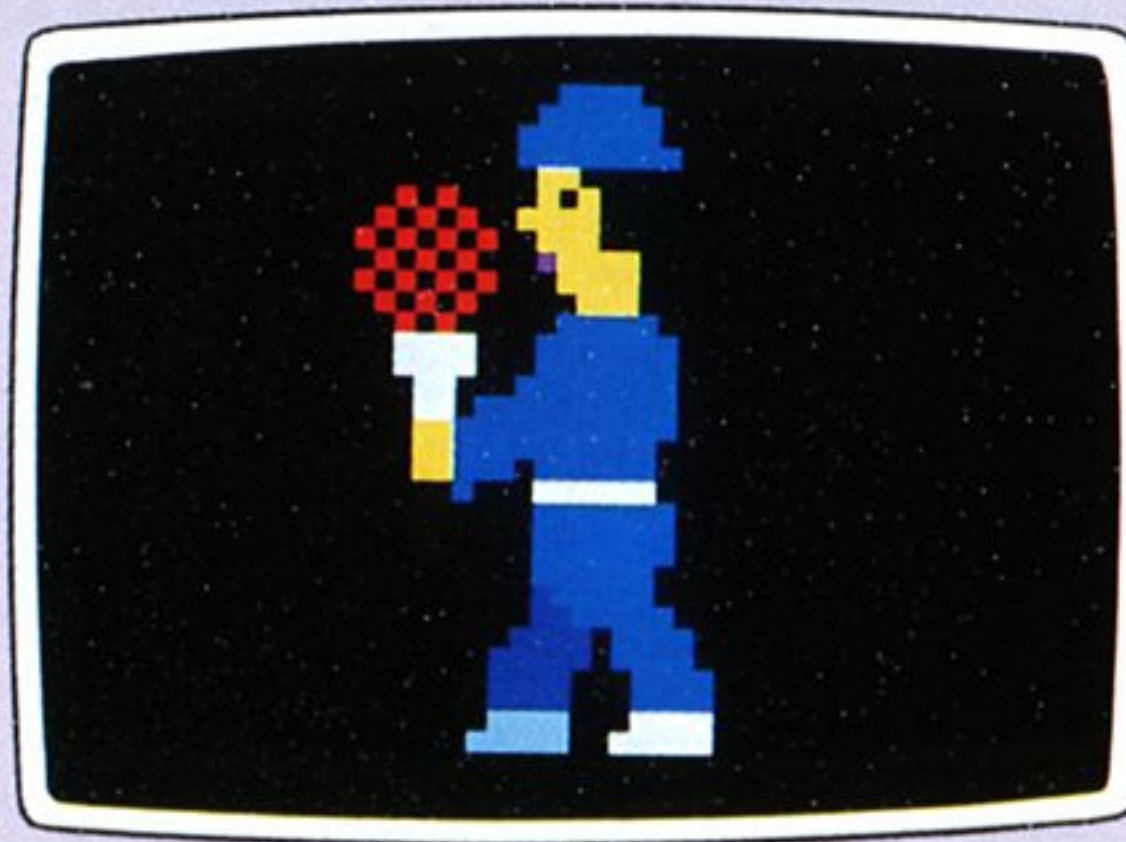
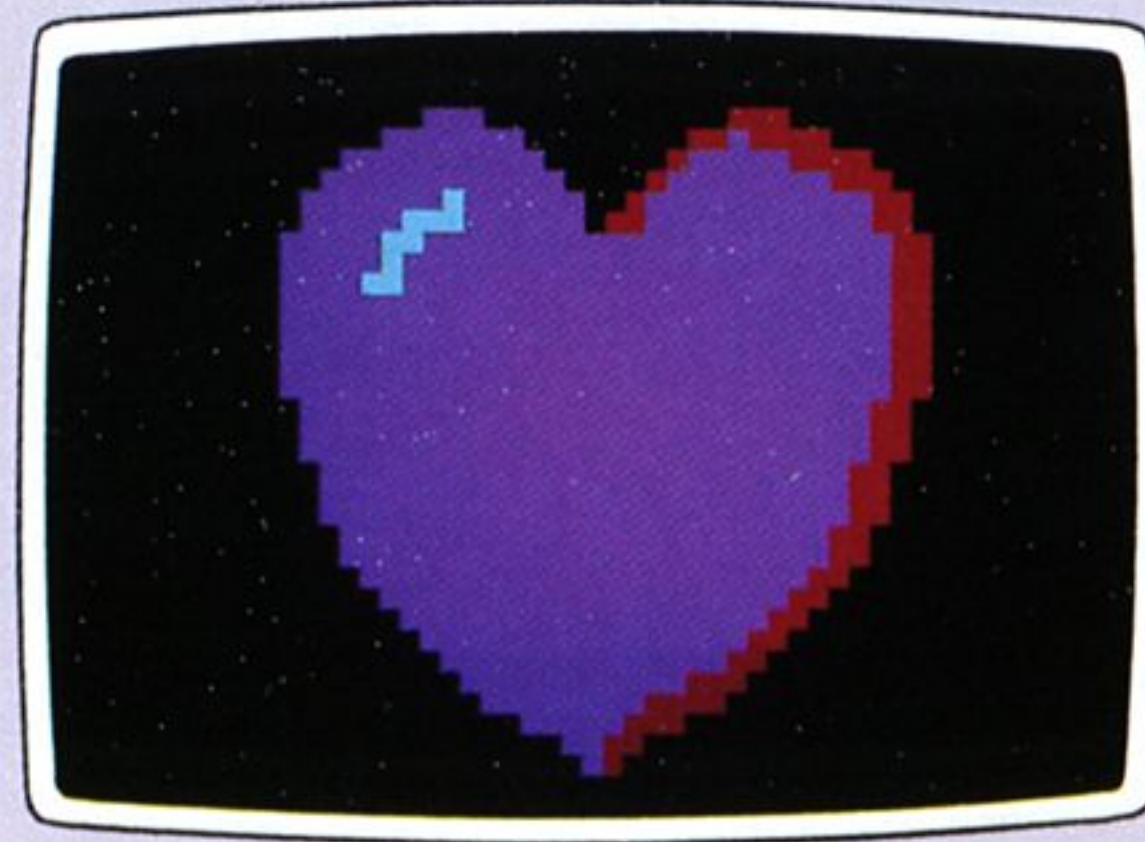
ピンク地，左足上げ(p.136,145)

緑地，右足上げ(p.136,145)

水面(p.136)

草原(p.136)

木(p.136)

城，左下(p.136)

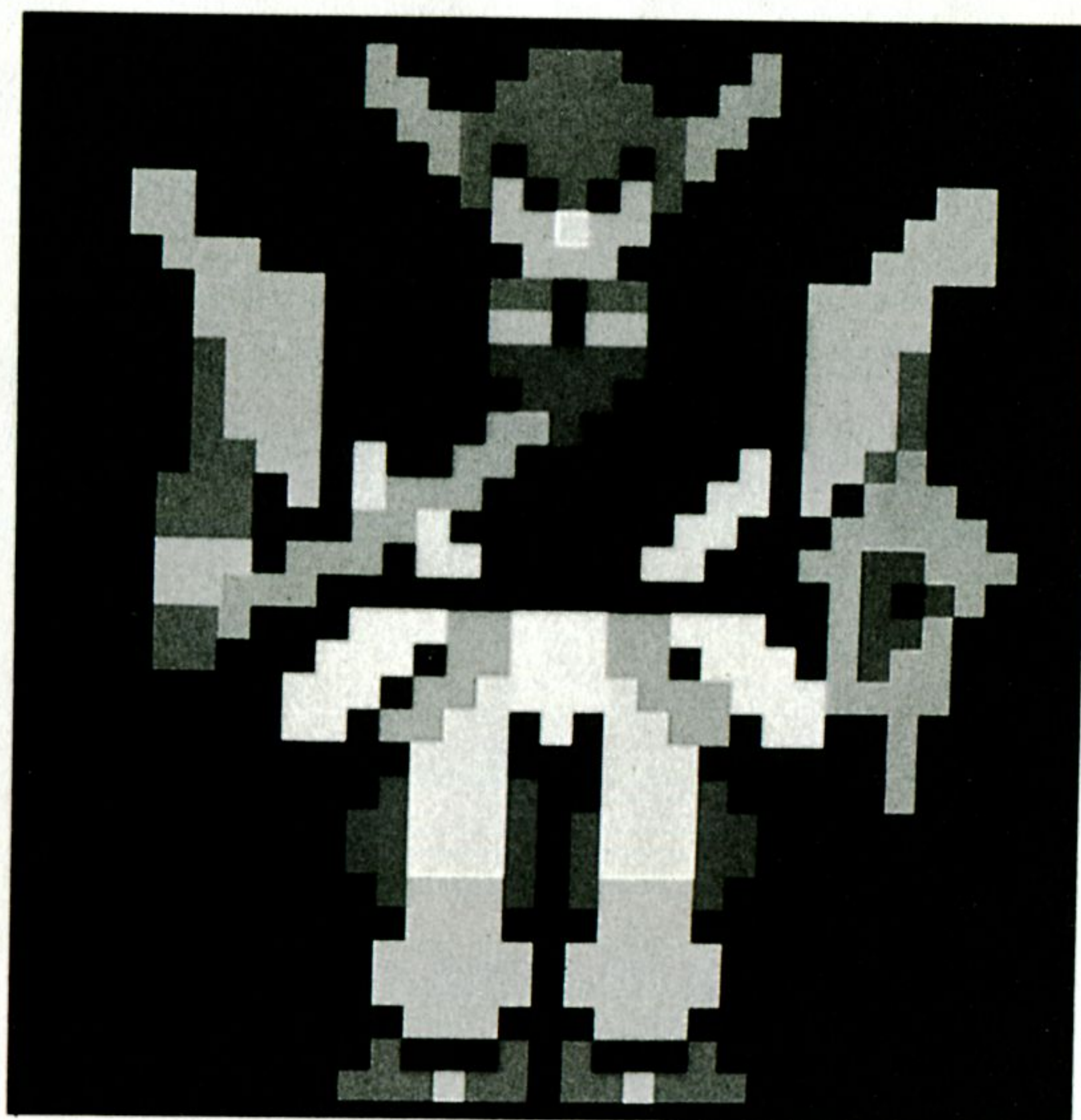
城，左上(p.136)

城，右下(136)

城，右上(p.136)

城，合わせると(p.136)

UFO小爆発(p.160)

UFO中爆発(p.160)

UFO大爆発(p.160)

UFO超大爆発(p.160)

味方ミサイル(p.160)

隕石(p.160)

敵の攻撃(p.160)

☆『プレゼント』アニメ

『プレゼント』アニメ(p.182)

花嫁(p.183)

花婿直立(p.183)

花婿左足前(p.183)

花婿右足前(p.183)

ハート(p.183)

ゲームプログラムでマスター Quick BASIC

PC-9801対応

稲田浩一朗 著

誠文堂新光社

は じ め に

○どうも，いままでの入門書では難かしすぎるし，つまらないと思っている人.
○せっかくコンピュータを始めたのに自分のやりたいことがうまくできない人.
○市販のゲームじゃおもしろくない人.
○ゲームプログラマーの仲間になって，大金持ちになりたい人.
○コンピュータを使ったプレゼンテーションをやりたい人.
○子供達に，楽しくコンピュータを教えてあげたい人.

この本は，そういう人のために書かれました．いま話題のQuickBASIC(クイック ベーシック)を使ってスーパーゲームを題材に，なるべく楽しく，わかりやすくコンピュータのプログラムを説明しています.

また，子供だましのゲームではなく，市販のゲームに負けないようなさまざまな本格的なゲームの作り方についても学ぶことができます.

コンピュータを前に，実際にプログラムを動かしながら読み進んでいくことをお勧めします.

読み終えるころには，知らず知らずのうちに，プログラムに関するさまざまな知識と，いろいろな分野に応用できるテクニックが身に付くことと思います.

最後に，本書の出版に当たり，多大なご尽力をいただいた誠文堂新光社の北村俊一様，および川名昭治様に心からお礼を申し上げます.

1992年7月　　　　著　　者

目　次

コラム

本書の使い方

本書のプログラムは，MS-DOS（エムエスドス）と Quick BASIC バージョン4.5を使用しています．また，使用したパソコンは，PC-9801DA-2 および PC-9801SX/E です．ハードディスクはあった方が楽ですが，なくても構いません．

速度などが，いま使っているパソコンに合わないときは，途中で時間かせぎに使っている，

```
FOR_I_=_0_TO_?
```

の，?にあたる部分の数字を適当な値に変えてください．キャラクタを入れるための配列の値は，他のパソコンでの使用も考えて，あらかじめ大きめにとってあります．IBM PC/AT，IBM PS55 シリーズ，FM TOWNS，FMR シリーズ，アップルのマッキントッシュシリーズなど，Quick BASIC の走る他のパソコン上でも，SCREEN 命令のあとの数字，画面セーブをするときの番地など，若干の変更でプログラムは走るはずです．著者の場合，同じプログラムを，

```
SCREEN_12:WINDOW_SCREEN_(0,_0)-(639,_479)
```

とするだけで，VGA 対応の IBM PC/AT 上で走らせています．パソコンの性能が上がってきた現在，大切なのは速度などよりも移植性だと考え，機械語（マシン）の使用はしていません．(プログラム中の，_ はスペースの意味です)

サブルーチーンは，本体のメイン部分と分けた方が，メモリ効率や他のプログラムへの応用という点で優れているのですが，管理部分を本の中に書くとわかりにくくなるので，昔のまま本体のプログラムの中に書くやり方をとっています．

本書に掲載のプログラムは，PC-9801シリーズVX21，RX，RA，DA…，およびIBM VGA対応機などの16色カラー機に対応しています．8色モードしか使用できない機種をお持ちの方は，適宜プログラムを変えてください(90ページのコラム参照)．また，初代PC-9801，E，F，M，Uなどの機種は音楽の演奏ができないものがあります．

また，Quick BASIC自体のディスクへのインストールなどについては，スペースの関係上，掲載してありません．マニュアルを参照するか，または他の入門書を参考にしてください．

★注意書のマーク

本書中の注意書には，下のような意味のマークが付いています．

：注意してください．

：知っておくと便利です．

：残念ながら，場合によっては満足のいく結果が得られません．

★登録商標について

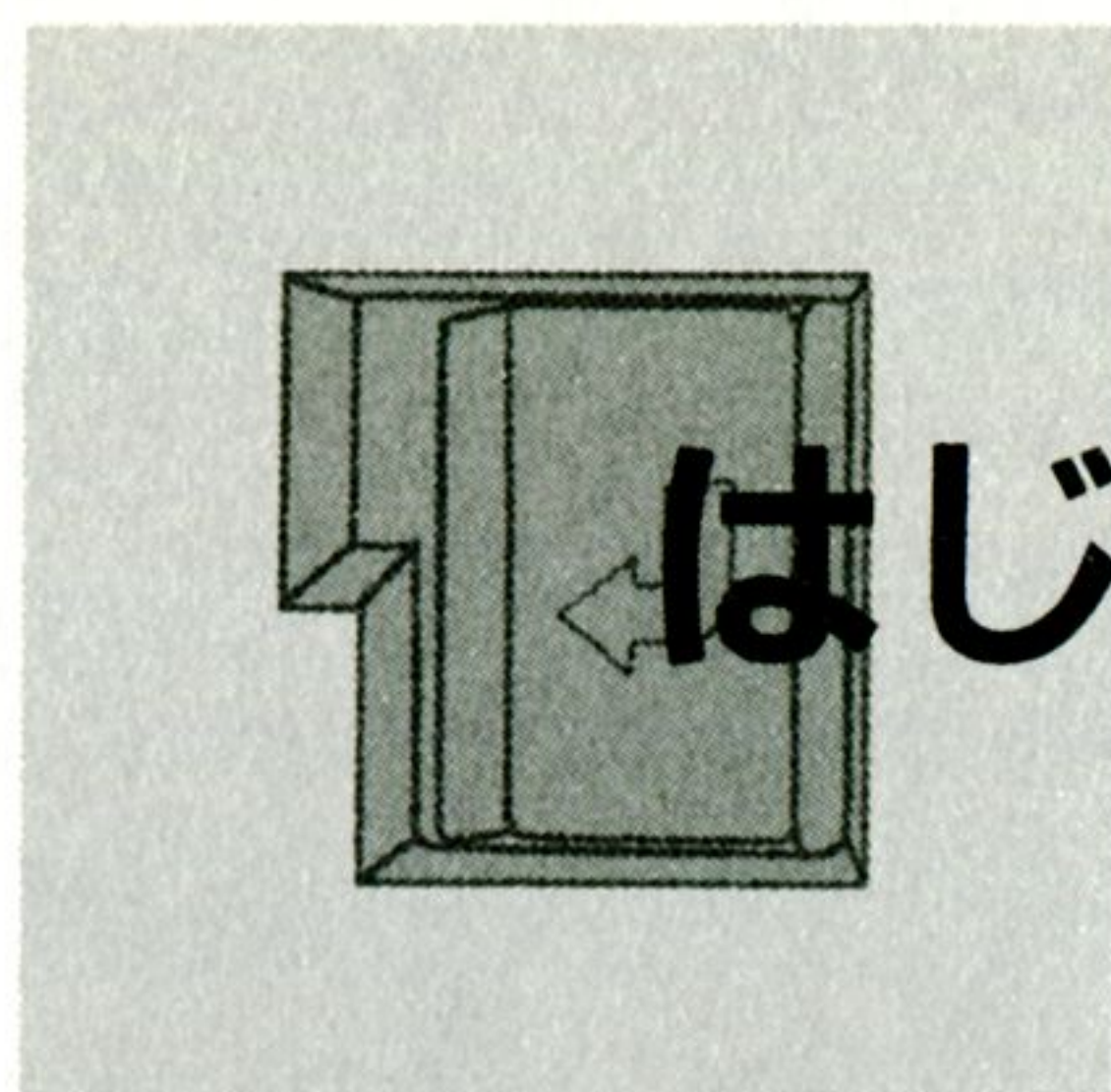
○Microsoft，MS-DOS，Quick BASIC，Microsoft Word，Word BASIC，Visual BASIC，Windows，EDLINは，米国マイクロソフト社の登録商標です．

○PC-9801は，日本電気(株)の登録商標です．

○IBM PC/AT，PC-DOSは，米国IBM社の登録商標です．

○マッキントッシュは，アップルコンピュータ社の登録商標です．

○FM TOWNSは，富士通(株)の登録商標です．

○その他，各種市販ソフト名は各社の登録商標です．

はじめの注意！

プログラムの勉強で大事なのは，まずやってみることです．

本を読むだけではなく，ぜひパソコンの前に座って，実際にパソコンを動かしながら覚えていってください．

プログラムを入れる(入力する)ときには，いくつか注意することがあります．実際に始める前に，次の『プログラムを入力するための注意事項』をお読みください．

★プログラムを入力するための注意事項

本書に掲載されているプログラムをパソコンに入力するときには，次のようなことに気を付けてください．

１つ１つの文字は，固有の意味を持っています．よく似た文字に注意して，正確に入力してください．

とくに間違いやすい文字としては，次のような文字があげられます．

カンマとピリオド	,	.
コロンとセミコロン	:	;
クォテーションとダブルクォテーション	'	"
数字のゼロとアルファベットのオー	0	O
数字の１と英字小文字のｌ (エル)	1	l
数字の３と数字の８	3	8
マイナス記号とアンダーライン	-	_
マイナス記号とカタカナの伸ばす音	-	ー

本書では，本文中のプログラム入力の説明において，空白（スペース）を表すのに_(アンダーライン)を使用しています．このアンダーラインのところでは，スペースキーを押して空白を入れてください．

また，プログラム中，0(ゼロ)とO(オー)は非常に間違えやすいので注意して入力してください．

シフトキーを押さなくてはいけない場合，気を付けないと，入力しようとした文字とは異なった文字が入力されますので注意してください．

長いプログラムを入力するときは，1行または数行をまとめて抜かしていないか，よく注意しましょう．

本書に掲載のプログラムにおいて，文字数の関係で1行に入らない行は，次行の先頭へ出して掲載しています(とくに，IFやDATA文など)．入力の際は，必ず1行に続けて入力してください．

また，1番最初の全角文字の1行はプログラムの名称です．入力しないでください．

単語と単語の間のスペースは重要です．続けて書かずにキチンとスペースをあけましょう．

×　　PRINTA

○　　PRINT_A ――→ PRINT A

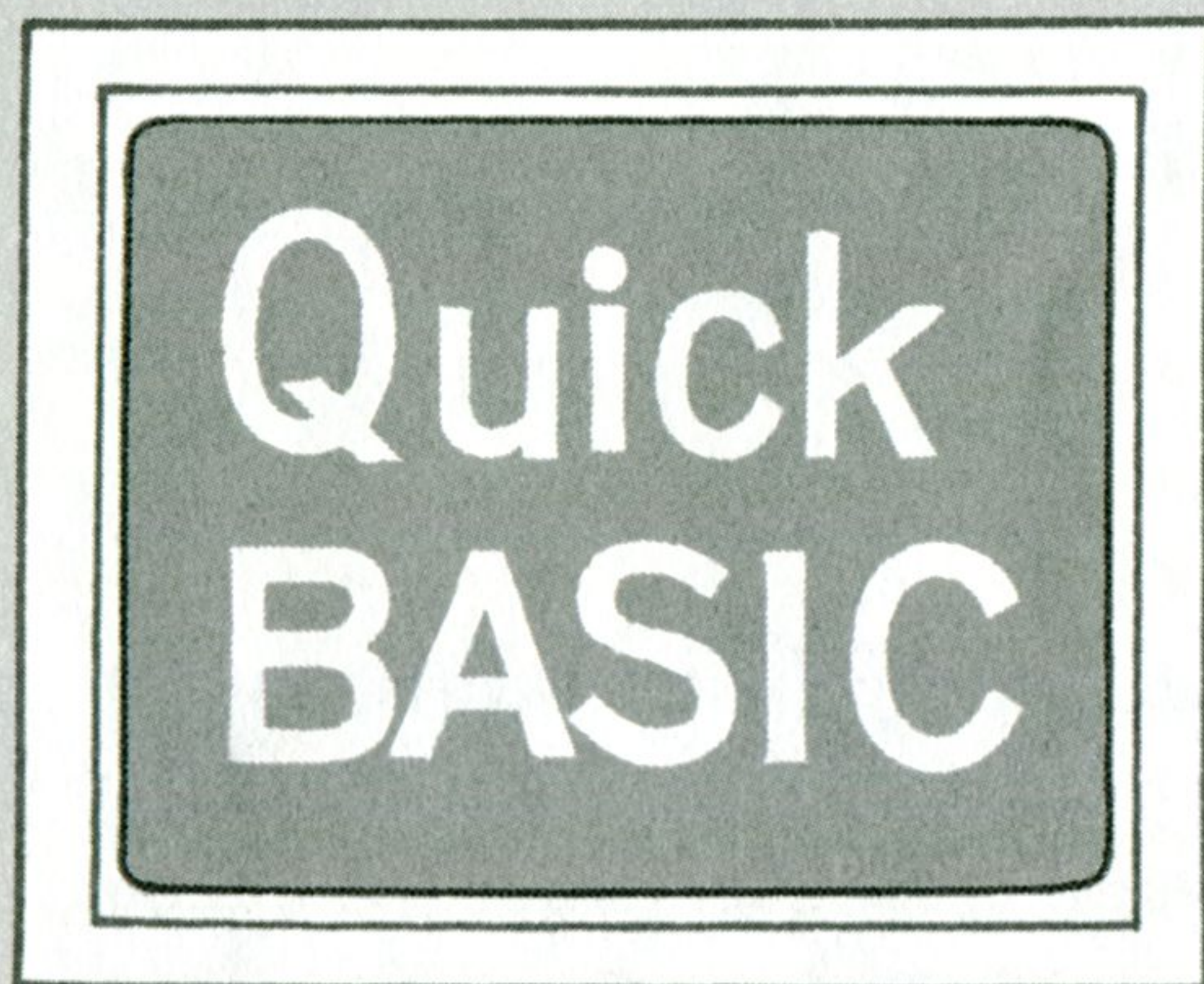
新しいプログラムを入力する前には，必ずBASIC操作のファイルの新規を選んで，古いプログラムを消してから初めてください．

少し入力したら，必ず保存しておきましょう．こうすることで，不意の停電など，さまざまな場合に対処できます．

プログラムを実行する前に，必ず保存しておきましょう．無限ループに入ったり，暴走したりしてリセットキーを押さなくてはいけなくなっても，プログラムは残ります．

実行してみて，エラーが出たら，まずエラーが出た行に間違いがないかをよくチェックします．エラーの出た行に間違いがない場合，そこから前に間違いがある確率が高いので，1行ずつ，そして1文字ずつ確認していきましょう．

PC-9801シリーズの8色モードしか使用できない機種については，そのまま掲載してあるプログラムを入力したのでは動かないことがあります．11ページの囲みの中，および90ページのコラムを参考にして，適宜プログラムを変更してください．

すばらしい
Quick BASICの世界

Quick
BASIC

いま，時代は Quick BASICなのだ！

昔のBASICは構造化ができない，行番号に依存する，速度が遅いなど，悪口をたくさんいわれました．でも，Quick BASICの出現で，それらの悪口はすべて過去のものとなりました．

いま一番ホットなBASIC言語，それがQuick BASICなのです．ここでは，まずQuick BASICのすばらしさについて見ていきましょう．

★マイクロソフトの大戦略

皆さんよくご存知のように，Quick BASICはマイクロソフト社が開発した言語です．この会社は，Quick BASICのほかにも，MS-DOS，IBM版のDOSであるPC-DOS，いま話題のWindowsなど，次々に優秀なソフトウェアを提供し，アメリカのパソコン界はマイクロソフト社のソフトウェアなしでは過ごせないところまできています．

マイクロソフト社の会長はビル・ゲーツです．彼は，彼の作ったBASICによって一躍有名になり，マイクロソフト社を設立したのです．そして，BASICの大好きな彼は，世界中のプログラムをBASICを中心にしようという大戦略に出たのでした．

IBM PC用のDOSのバージョンは，アメリカでは5.0です．このDOSには，何と，Quick BASICのサブセットが無料で付いています．また，このDOSに付いてくるエディタは，あの“メンドウクサイ”EDLINなどではなく，Quick BASICのエディタとまるで同じ使い方ができるものです．いかにマイクロソフト社が，Quick BASICに力を入れているかがわかります．

★ Windowsって知ってますか？

いま，アメリカでは，Windows（ウィンドウズ）と大さわぎをしています．

Windowsを使えば，メモリの640KBの制限など関係なく，目いっぱいメモリが使えます．また，メモリが足りなければ仮想メモリが使え，複数のプログラムを同時に走らせて，その間を簡単にいったりきたりできます．さらに，Windows用のプログラムは，パソコンの機種に関係なく走らせられるなど，さわぐのも当然です．

ところが，このWindows用のプログラム，使う方は楽なのですが，書く方は非常に大変でした．そこで，マイクロソフト社では，Visual（ヴィジュアル） BASIC（ベーシック）を世に出したのです．

★ Visual BASIC

Visual BASICは，マイクロソフト社がWindows用プログラムの開発のために売り出したQuick BASICに良く似た言語です．

Visual BASICを使うと，だれでも簡単にWindows用のプログラムが作れてしまいます．

いままでは，プログラマが，CやC++などを使って，ものすごく長いプログラムを書き，かつそれがなかなか動かず，大変苦労をしていました．

それに比べ，このVisual BASICを使えば，ウソのように簡単に，アイコンやメニューを持つWindows用のプログラムが書けてしまいます．

Quick BASICを知っていれば，Visual BASICも簡単です．

★ Quick BASICは最高級言語（CもC++も目じゃない）

昔々，BASICは，構造化ができない，速度が遅い，行番号に依存してプログラム部品の再利用ができない，などと悪口をたくさんいわれていました．でも，Quick BASICになって，上記のような欠点は克服されました．

Quick BASICの特長をいくつか上げると，

○構造化がしっかりできる．

○行番号に依存しない．

○サブルーチーン，ファンクションなどが，単独の部品として再利用できる．

○統合化環境が使える．

○インタプリタの手軽さと，コンパイラの性能の両方を満足できる．

○ DOSから直接実行できる実行形式の .EXEファイルを作ることができる.

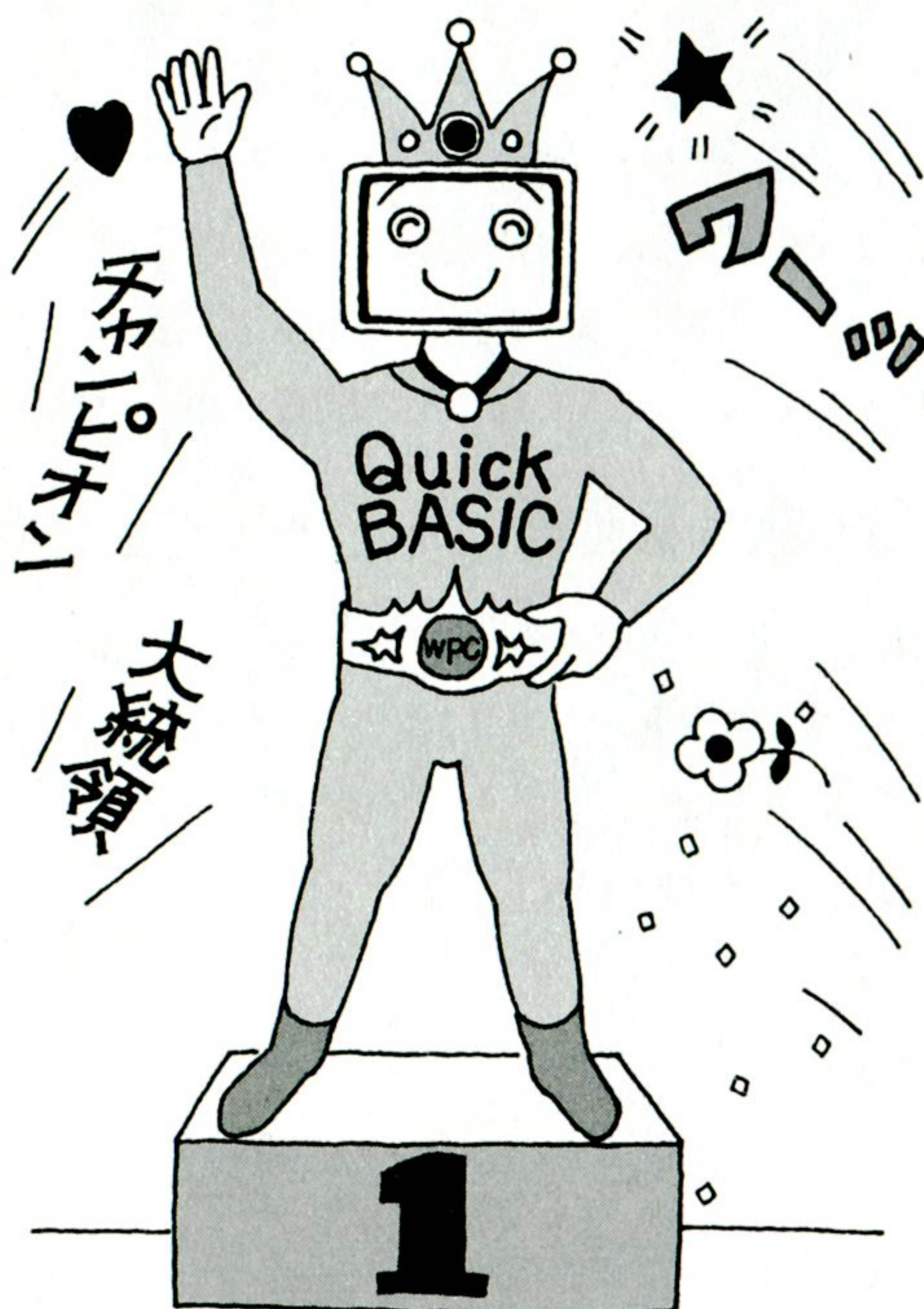
○使用できるメモリが飛躍的に増えた.

などが上げられます.

簡単にいうと，使い勝手が良く，やさしく高度なことができ，性能も良いということになります.

いま，Cというプログラム言語がブームですが，Quick BASICは，Cなどよりはるかに進んだ最高級言語なのです.

画面の操作，通信，音楽などを行わせるのに，Quick BASICでは命令(コマンド)1語ですむのに，Cなどほかの言語では何行もの式をかかなければなりません．これらの式をあらかじめ書いたグラフィックなどのパッケージもありますが，プログラムの書き方が各社ごとに違っていたり，パソコンごとに違ったりします．また，場合によっては，ある特定の動作をさせるためには，特定のメモリ上の番地を探し出し，そこに何かを書き込まなくては，うまく目的を達成できないこともあります.

CやC++は，高級アッセンブラではありますが，低級言語なのです.

★ 1＋1.1は，Cにはわからない

CやC＋＋では，ある式を使うとき，その式は変数の型に依存します．つまり，同じA+Bという式でも，1＋1と1.0+1.0では，まるで別の式なのです．Cでは，あまりに面倒だったので，C＋＋では，ある宣言をすると同じA+Bという式を，1＋1と1.0+1.0どちらでも使えるようにしました．それでも最初にいちいち宣言しないとエラーが出ますし，まして，1＋1.0などということはできません．

Quick BASICでは，1＋1.23などとやっても，パソコンが整数1は実数では1.0だから，これは2.23だなと，しっかり計算してくれます．

CやC＋＋などより，はるかに進んでいると思いませんか？

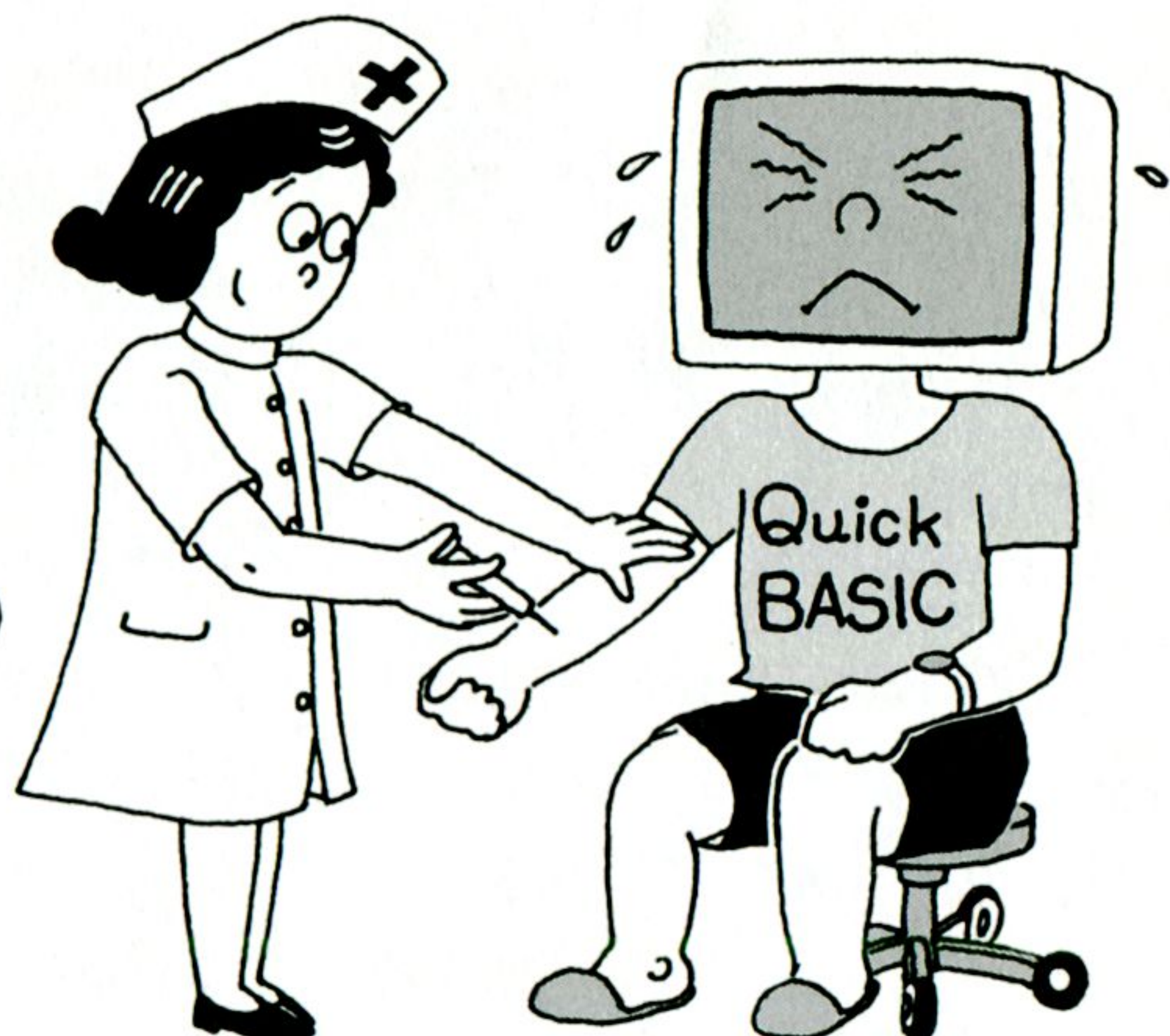
★ Quick BASICの血液型は何型か？

「エッ！　プログラムに血液型があるのか？」ですって？

それがあるのです．

血液型には，普通，人間の場合A，B，AB，Oとありますが，このうち一番有利なのはAB型です．なぜかというと，いざとなればどの血液型からでも輸血してもらえるからです．

ズバリ，Quick BASICの血液型は，AB型なのです．

Quick BASICの
血液型は？

そして，CもPASCAL（パスカル）も，みんなQuick BASICの子分です．

マイクロソフト社の言語は，お互いに相互利用が可能です．つまり，あるやり方をすると，BASICからCやPASCAL，FORTRAN（フォートラン），マシン語などのルーチーンが使えます．また，この逆も可能です．

ただし，BASIC特有のグラフィックやサウンドなど，BASICで一番使いやすい命令は，他の言語では使えないのです．

BASICで一番の特徴は，グラフィックやサウンドなどの入出力の命令が簡単に一語の命令で書けるところにあります．そこで，一番有利な方法は，プログラムの主な部分をQuick BASICで書いておき，細かいメモリ操作，高速性などの理由で，どうしても他の言語を使う必要のある部分だけを他の言語で作ったり，借りたりしてくる方法です．

このように，これからプログラミング言語の中でQuick BASICは，その使い勝手の良さ，構造化，他言語への対応，そして，簡単に他の種類のコンピュータに移植できることなど，時代の主流を歩むのは間違いありません．

いまやC言語では時代遅れなのです．

★マクロって知ってますか？

「エ～ト，某ニックキヤツの根性のことでは…」

「エッ，違います．それって，真っ黒のことでは？」

マクロというのは，プログラムを便利に使うために書く短いプログラムのことでも思ってください．ワープロソフトから表計算ソフトまで，市販のソフトウェアはよくできているのですが，どうしても１人１人の好みに完璧に合うようには作れません．また，場合によっては，同じように決まりきったことを毎回やらねばならず，それを，いつもいつもキーボードから入力していたのではとても大変な場合があります．

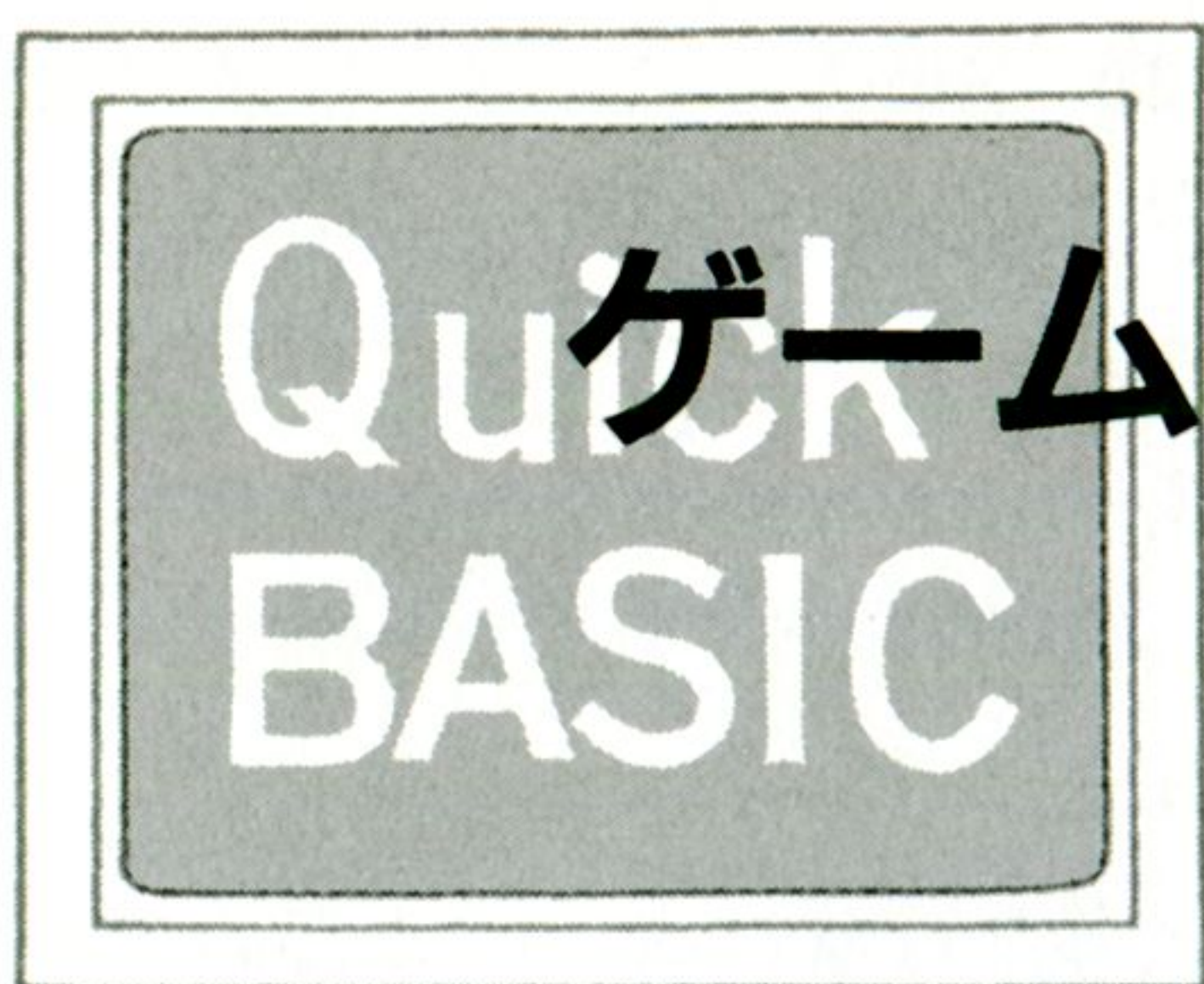
そこで，そういう場合に大活躍するのがマクロ言語です．マクロ言語でプログラムを書いておくと，ワープロでも表計算ソフトでも，“あなた好みに変身”してくれます．おまけにソフトウェアによっては，決まりきった処理はマクロプログラムを書くことで自動的にやってくれます．本当にとっても便利なのです．

ところが，このマクロ言語には1つだけ大きな欠点があるのです．いままでのほとんどのソフトウェアに付いてくるマクロ言語は，&とか#とか記号の羅列なのです．さらにもっと悪いことには，こっちのエディタとこっちの表計算ソフト，また，あちらのワープロと，お互いに好きかってにマクロ言語を作ったため，まるっきり文法が違います．これではいかに便利なマクロでも困ってしまいます．そこで，マイクロソフト社では，自社で作るさまざまなソフトウェアのマクロ言語をQuick BASICに似た新しいマクロ言語に置き換えています．

マイクロソフト社からMicrosoft Wordという日英両用のワープロソフトが発売されました．これは，次世代のワープロとして注目されているワープロソフトですが，これにはWord BASICという新しいマクロ言語が付いてきます．これを使えば，ワープロソフトのプログラム本体を修正しなくても，まるで本体を修正したかのように高度なことがスラスラできてしまいます．自分好みに機能が拡張できるというのは楽チンですネ．

このWord BASICは，名前のとおりQuick BASICにとっても良く似ています．いままでの記号ばっかりのマクロ言語とは大違いです．同社では，これからほとんどのプログラムのマクロ言語をBASICに似た同じようなマクロにしていくことを宣言しています．これによって，マクロを覚えるのがとっても簡単になります．また，ワープロのマクロで覚えたこととほとんど同じ感覚で表計算ソフトのマクロが書けます．これらのマクロ言語すべての基本となっている言語がQuick BASICなのです，『Quick BASICを制するものはプログラムを制する』といっても過言ではありません．

マクロ言語のことを考えただけでも，Quick BASICは習っておいて損にはなりません．

Quick BASIC

ゲームを作って友達をうならそう

★ゲームを通してプログラムを覚えよう

いま，町にはスーパーファミコンからPC-9801まで，いろいろなパソコンやゲームがあります．

でも，ゲームをやりながら，ここをこう直したいとか，こういうゲームを作りたいと思ったことはありませんか？

自分でオリジナルのゲームを作って，友達から「スゲェー，どうやって作ったんだ」と，驚きと尊敬のまなざしで見られたら，最高にカッコイイと思いませんか？

この本を読みながら実際にパソコンを使って行くと，ゲームを作りながら，プログラムのやり方やゲームに必要なさまざまなテクニックが楽しく学び取れ，知らず知らずのうちにプログラムが作れるようになります．

★ゲームを教えて，お父さんの株はストップ高

子供達にとって，お父さんはいつでもヒーローです．

そのお父さんが，子供達の大好きなゲームを作ったとしたら，まして，その中で自分が作って名前を付けたキャラクタが動き回ったとしたら，子供達にとって，お父さんがスーパーヒーローになるのは間違いありません．

家族そろってパソコンを前に，「今度はここを直そうよ」とか，「この方がおもしろいよ」と話をしているなら，親子の断絶なんて，絶対にありえないでしょう．

★小学生でもゲームが作れた

筆者の家にも，小学生の子供が２人います．彼らにとってはパソコンは難しい機械でもなければ，命令を覚えなくてはならない機械でもありません．そう

です．パソコンはお友達なのです．また，自分達のやりたいことがいろいろできる夢の箱なのです．

最初のうちは，ファミコンに，それにあきるとPC-9801などのパソコンのゲームで一生懸命遊んでいた子供達でしたが，そのうち，「お父さん，僕達にもゲームの作り方を教えてよ」といってきました．「いいよ」といって，最初は，いわゆる入門書どおりに教え始めたのですが，これは大失敗でした．なんせ型どおりの命令PRINTなどからでは，おもしろくもなんともありません．そこで方針を大転換，まず楽しさを中心に，動きと音から始めたら，今度は大正解．あっという間にプログラムを覚えてしまいました．

本書の中のプログラムの一部は，実は彼らが作ったものなのです．いまでは，市販のゲームで遊ぶより，自分達のプログラムをいじっている方が楽しそうです．

★ゲームを売って億万長者

自分が作ったゲームが完成すると，友達にやらせて見たくなります．人によって，いろいろな意見があると思いますが，そこはしっかり聞いて，どんどん改良しましょう．

ゲームのレベルが上がって，いいものができたと思ったら，ぜひゲームを売りに行きましょう．あの“ロードランナー”を始め，さまざまなゲームが，この方法で出きました．プログラマの中村光一氏も，最初はゲーム“ドアドア”を作って，高校生の時に2000万円以上(？)ももうけてしまったそうです．その後，“チュンソフト”という会社を作り，なんとあの“ドラクエシリーズ”のソフトウェアなどを作っています．いまや，そのプログラムの著作権料は，1つのシリーズごとに数億円(？)だそうです．そして，若くして押しも押されもしない大社長となりました．

マシン語でなくては売れるようなプログラムを作れないと思ったら大間違いです．ゲーム“倉庫番”はBASICで書かれています．それもデータ以外の本体のプログラムは，『こんなに短いの？』というぐらいの長さです．でも，このゲームは，他のゲームと一味違っていて，やり出すと夢中になってしまいます．ちなみに，このゲームを作った人も，脱サラして頑張っています．

“テトリス”も“上海”もルールはものすごく単純です．それでいて人を熱くさせるものがあります．

ゲームで大事なのは，アイデアとコンセプトです．あのシミュレーション・ウォーゲームとしてすごく有名な“大戦略”も，会社に売り込まれたときは，すべてBASICで書かれていたそうです．いいゲームで，キチンと書いてさえあれば，速度に関しては専門に直してくれる人達がいます．この人達の手にかかれば，超高速にしたい部分が次々にマシン語に変わってしまいます．

さあ，あなたもこの本でいろいろ覚えて，ぜひスーパーゲームを作りましょう．スーパーゲームができたら，ぜひ第2，第3の「若い大社長」になって，億万長者になりましょう．

プログラムまだ作れないんですか?

Quick BASIC

パソコンの性能は，日進月歩で進化しています．いまのパソコンの最高級機種の性能は，一昔前の大型コンピュータをはるかにしのいでいるのです．

一方，プログラムの方も，同じように進歩しています．いま一番最初のころの表計算ソフトや，ゲームソフトを見ると，よくこれで売れていたものだと感心してしまいます．でも，それらのソフトウェアも，当時としては画期的なものだったのです．

★プログラム，あなた作る人，私使う人

これからの世の中は，ソフトウェアの進化とともに，ソフトウェアを作る人と使う人に，はっきりと別れて行くことでしょう．

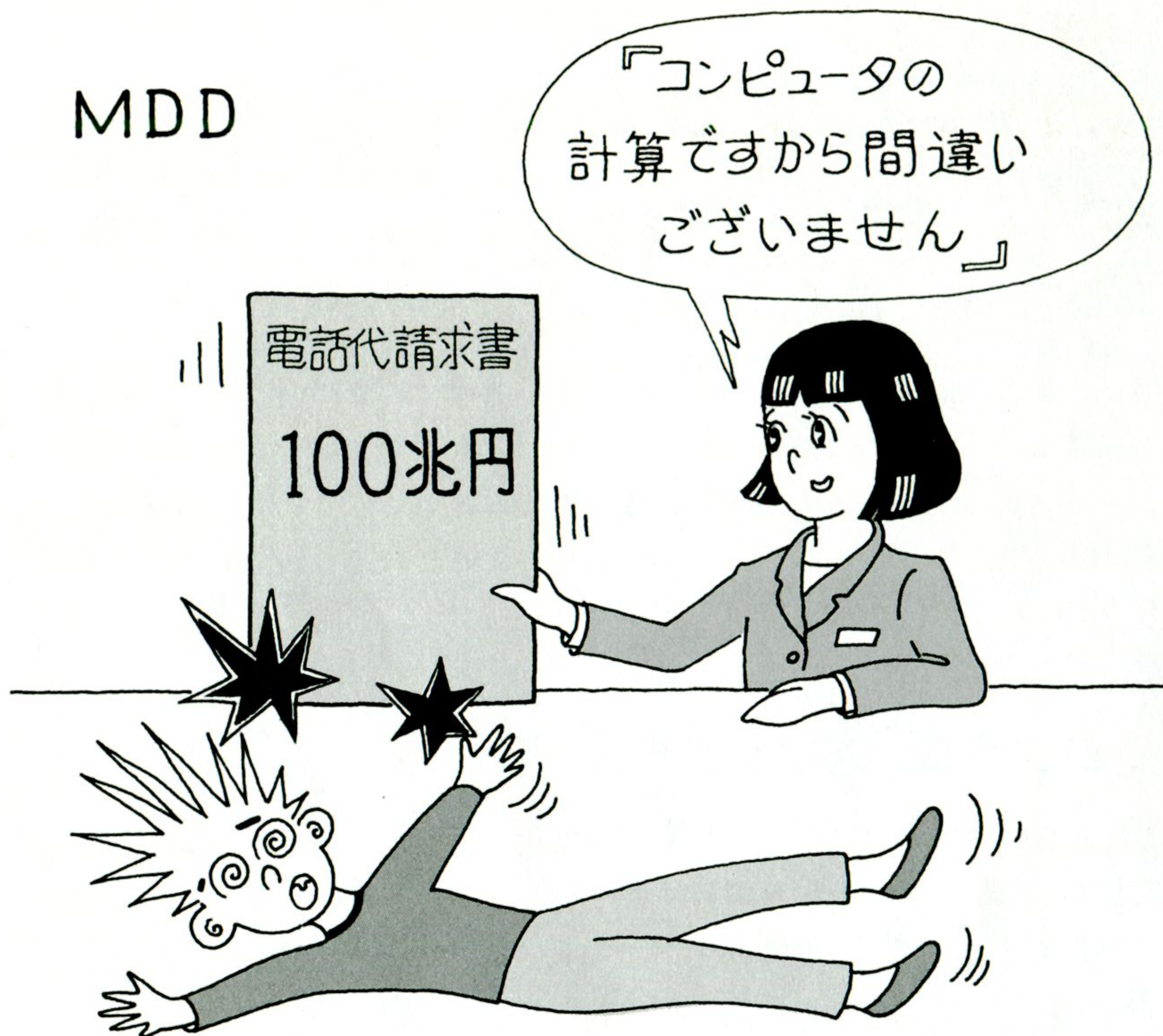
このままでは，ごく一部の人達がソフトウェアを作り，ほとんどの人は，ただそのソフトウェアを買って使うだけということになりかねません．

実は，これはとんでもないことなのです．昔，某電話会社が，コンピュータを導入したとたん，月間数百万円というとんでもない電話代を請求された人がいたそうです．電話会社に文句をいいにいくと，最初は「これはコンピュータが計算したのだから，絶対に間違いはございません」と，相手にしてもらえなかったとか….

プログラムを自分で作ったことのない人なら引き下がるしかありませんが，チョットでもやったことがあれば，「これは入力ミスか，プログラムのバグのせいではないのですか？　入力チェックのルーチーンはどうやっているのですか？」などと，相手を問い詰めることもできるでしょう．

MDD

『コンピュータの
計算ですから間違い
ございません』

電話代請求書
100兆円

★プログラムが作れないと，時代に取り残されます

これからの時代，コンピュータは必須科目です．

少しでも自分でプログラムの経験があれば，コンピュータのいろいろな特性が見えてきます．時代に取り残されず，コンピュータパワーエリートになるためにも，ぜひ，プログラムを勉強しましょう．

★秘密がわかれば簡単だい！

コンピュータのプログラムというと，とても難しそうですが，実はそんなことはまったくありません．だれかに「窓を開けてください」と頼むのと同じで，命令語と，それに対応する数字(データ)さえ書いてやればいいのです．

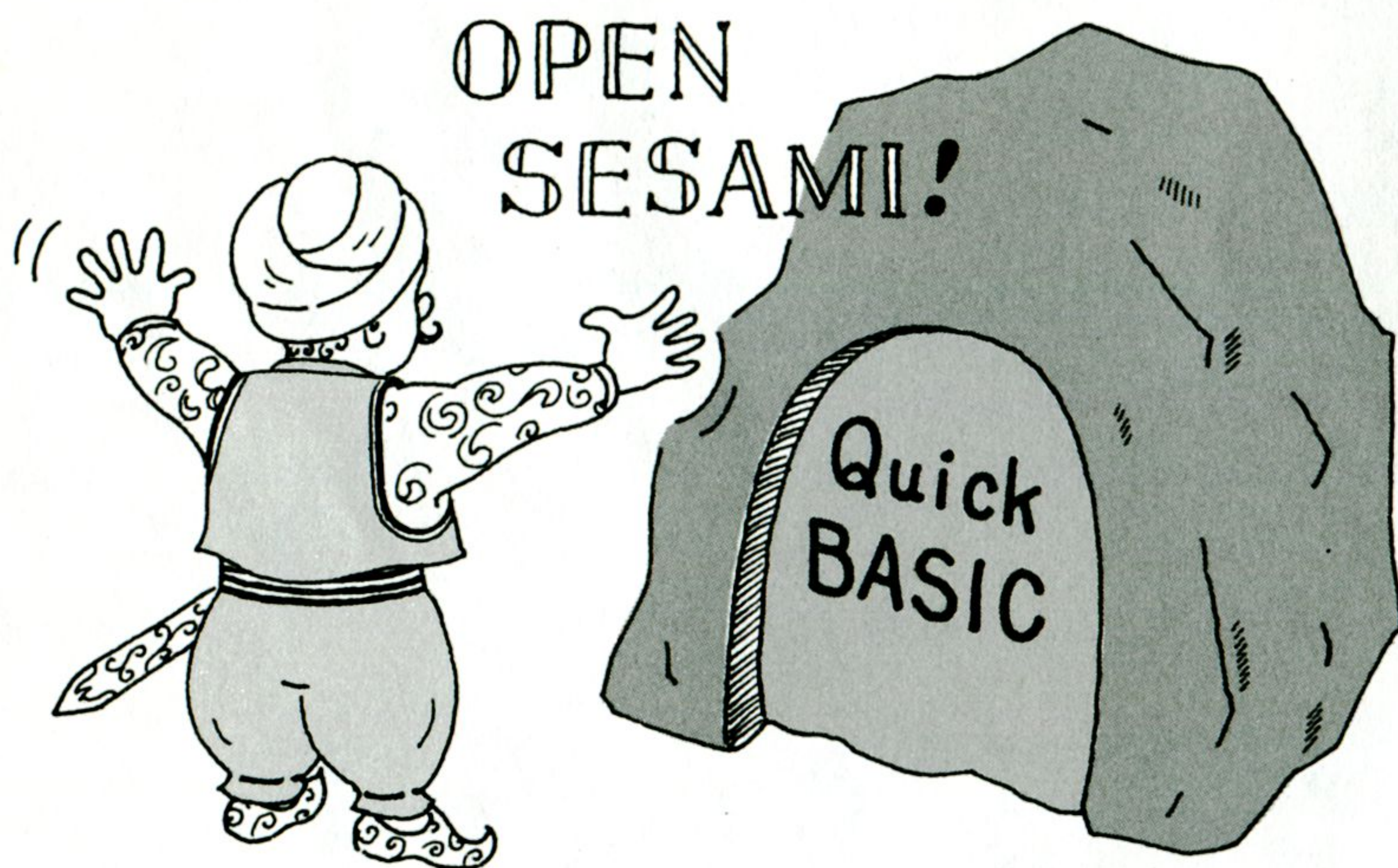
そうです，実は秘密さえわかれば，プログラムなんてとっても簡単なものなのです．

Quick BASICのコマンドや関数などを楽しく学ぼう！

Quick BASIC

開けゴマ，OPEN SESAMI！

世の中，なにかを始めるに当たっては，いろいろと儀式や呪文が必要なことがたくさんあります．さしずめ，一番有名なのが“アリババと40人の盗賊”に出てくる『開けゴマ』でしょう．ところで，この『開けゴマ』英語でいうと，やっぱり『OPEN（オープン） SESAMI（セサミ）』なのです(SESAMIはゴマです)．あのテレビ番組の“セサミストリート”のセサミもこの言葉が元なのだそうです．

では，Quick BASICの世界を開くために，皆で『OPEN SESAMI！』

32ページのプログラムの最初の3行を見てください．なにやら見なれない単語と数字が並んでいますが，この単語と数字がQuick BASICの世界で，絵を扱うための呪文なのです．とりあえずは，別に深い意味は考えず，絵を扱うときはこの3行と同じ文をプログラムの最初に書くということだけを覚えておいてください．

次の部分は興味のある方だけどうぞ．

とはいっても，知らないで使うのは気持ちが悪いというあなたのために，ひととおり説明しておきましょう．

まず最初の行の　SCREEN(スクリーン)_88,_3,_1,_1　ですが，まず最初の数字88は，横640ドット，縦400ドットの画面に，絵と文字を書くということをパソコンに教えています．VGAシリーズなど他のドット数の画面では，この数字が変わります．次の3は，使える色数を決めています．

0　　8色
1　　16色
2　　64色中16色
3　　4096色中16色

となります．ただし，実際に使える色は，ここで指定する数字だけでなく，パソコンとディスプレイの性能により異なります(90ページのコラム参照)．

この番号を0(ゼロ)にした場合，8色モードになります．

8色モードは，PC-9801E，F，M，VM 2，VF 2，UV，VM21，VX 2，VX21，RX，RA，DA…と，すべての98シリーズで使えます．

番号1，2，3は，VX21，RX，RA，DAなど，アナログディスプレイに対応した機種のうち，ROM(ロム) BIOS(バイオス)で16色出せる機種用です．

8色対応の機種を使っている人は，SCREEN_88,_0,_ としてください．

PC-9801は，絵をかく面を2面持っています．通常，この面の一方を0，他方を1として指定します．

後ろから2番目の数字の1は，アクティブページといい，どの画面に絵をかくかを示しています．

最後の1は，ビジュアルページといい，実際に目に見える画面を，どちらにするかを決めています．

これを上手に使うと，0面を表示している間に1面に絵をかいておき，かき終わったら1面を表示するというように，絵をかくところを見せずに，画面を変えることができます．このテクニックは，アニメーション，ゲームなど，いろいろな場面で使われています．

パソコンの性能に併せて，色の数に対応する数字を変えてください．

次に2行目の WINDOW SCREEN 文ですが，ここでは，画面を左上が0，0右下が639，399の番号を付けた640×400の点（ドット）で表す画面にして絵をかきますよということをパソコンに宣言しています．このやり方の方が，点の付け方がいままでの N_{88}-BASIC と同じなので，わかりやすいと思います．

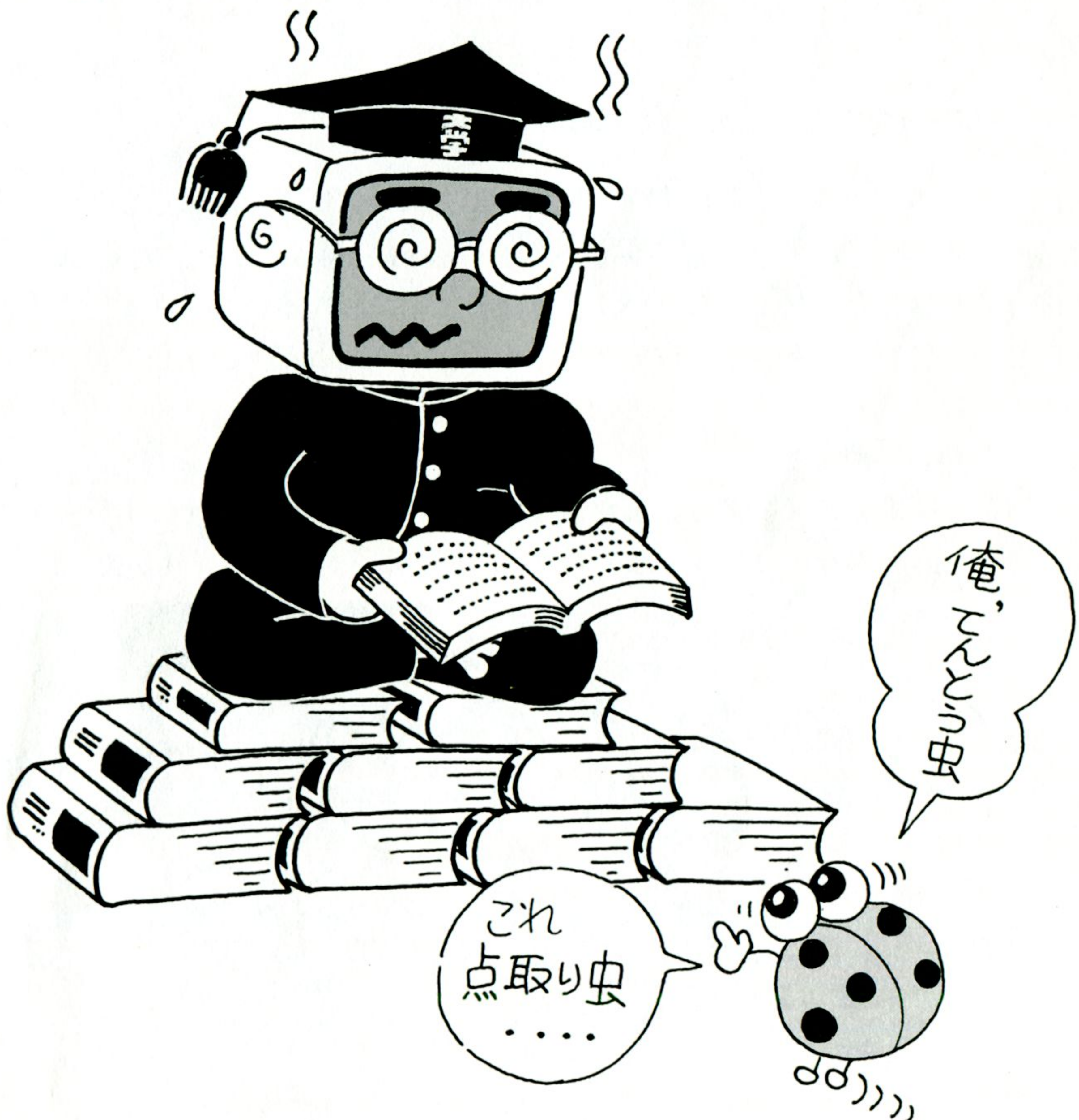
左下が0，0で，右上が639，399の方が都合がいいときは，SCREEN の文字を抜かしてください．

次に，3行目の CLS（クリアスクリーン） ですが，これは，それまでに画面に描かれているものを消してしまうという意味を持っています．CLS のあとに何が付くかによって，次のような違いがあります．

CLS	すべての文字と絵を消す．
CLS 0	すべての文字と絵を消す．
CLS 1	文字は残して絵だけを消す．
CLS 2	文字だけを消す．

Quick BASIC

点取り虫，点付け虫？

“点取り虫”オーッ！　なんと響きの悪い言葉でしょう．実社会でのこの言葉，どう見てもあまり好かれている言葉ではありません．実は，Quick BASIC の世界にも，この点取り虫ならぬ“点付け虫”が住んでいるのです．では，この“点付け虫”を探しに行きましょう．

次のプログラムを入力したあと，シフトキーを押しながらf・5キーを押して実行してみてください．

ＰＳＥＴ．ＢＡＳプログラム

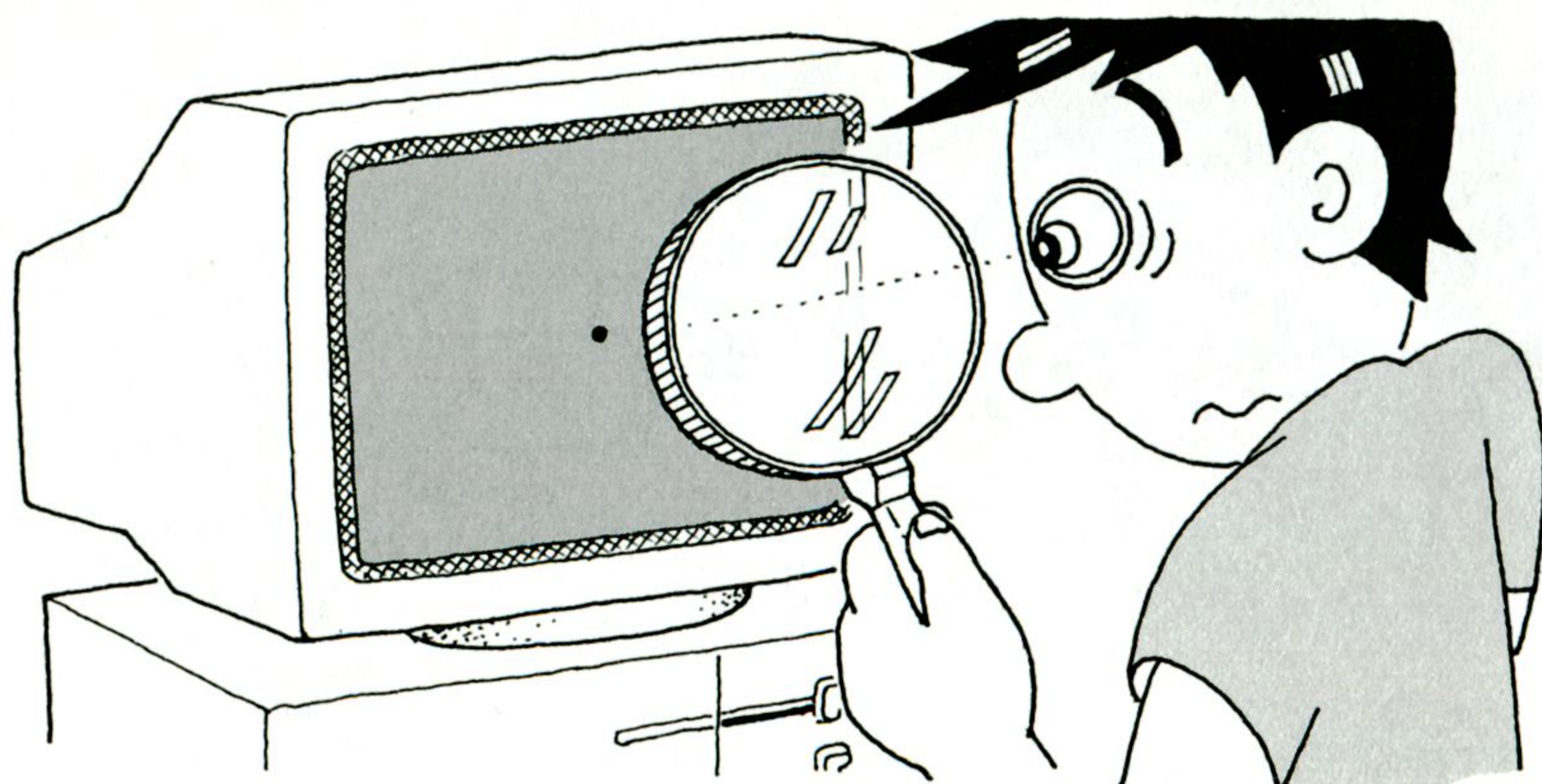
```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
PSET (320, 200), 4
END
```

なにも変わらないですって？　そんなことはありません．画面の中央をよーく見てください．なにやら小さな点が光っていませんか？　これが画面に点を光らせる命令PSET（ピーセット）の使い方なのです．ほーら，プログラムって簡単でしょう．もうあなたもプログラマーの仲間です．

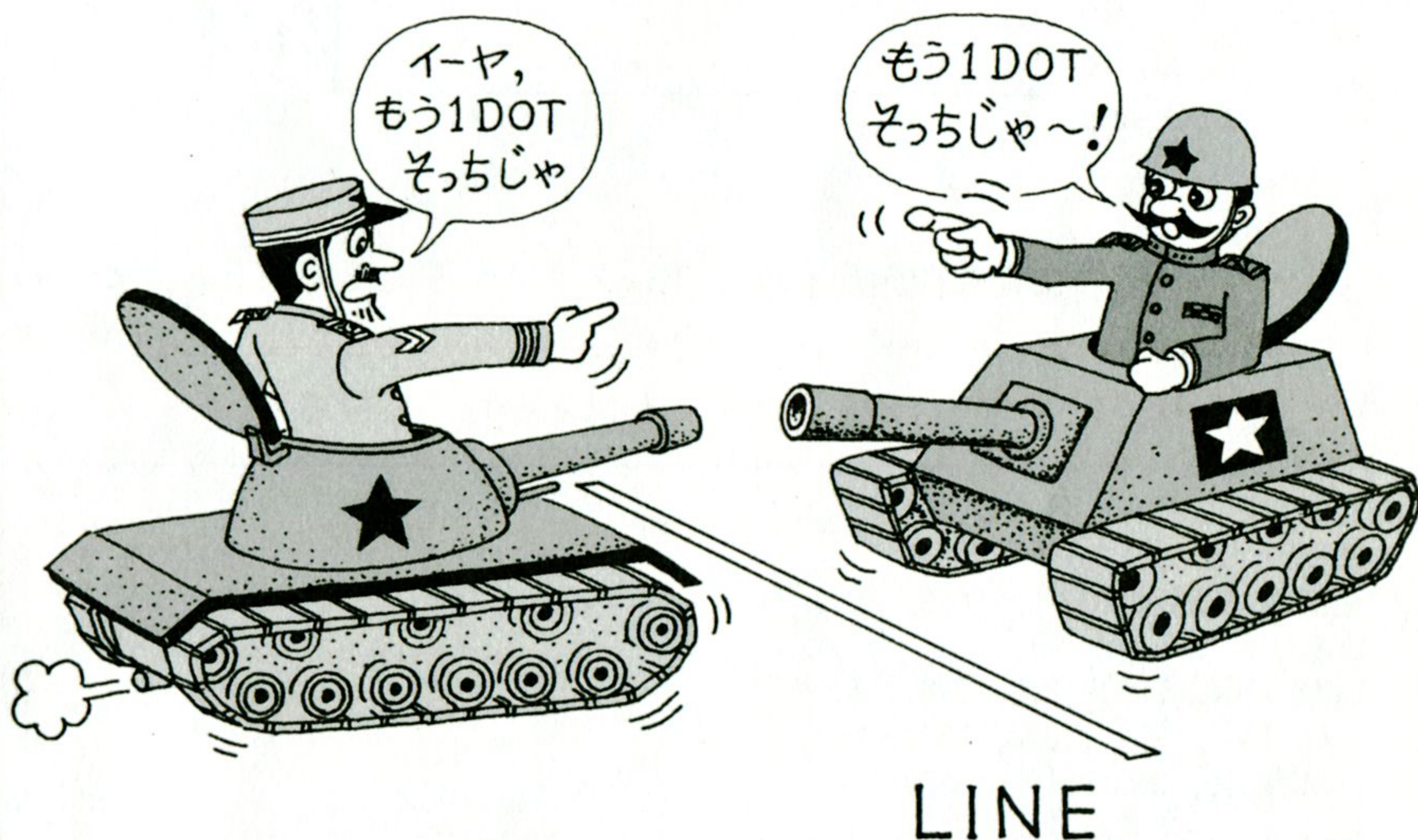
PSET_(横方向の位置,_縦方向の位置),_色

PSETのあとの数字は，横方向の位置や縦方向の位置，それに色などの意味を持ちます．PC-9801では，横方向は0から639，縦方向は0から399の値を持つ数字，または式を入れてください．色は，あとで説明するパレットというものがあり，どのパレットの色を塗るかを決めています．とりあえず色には0から15までの数字を入れてください．

なにかキーを押すと，エディット画面にもどります．

今度は，PSETの文のあとに書かれている数字をいろいろ変えて実行してみてください．どんなふうになりましたか？

Quick BASIC 一線をかくす？（LINE）

人間は，線を引くのが大好きです．大は国境線から，小は隣の家との境界線まで，なにもないところに勝手に線を仮想します．あげくの果ては，その仮想の線のために戦争を起こし，たくさんの人が死んだり，土地争いを起こして『殺人だ』，『裁判だ』と，バカ(?)なことをやっています．オーッコワ！

ここでは，そんな恐ろしい線ではなく，もっと楽しくハッピーな線の引き方です．

LINE

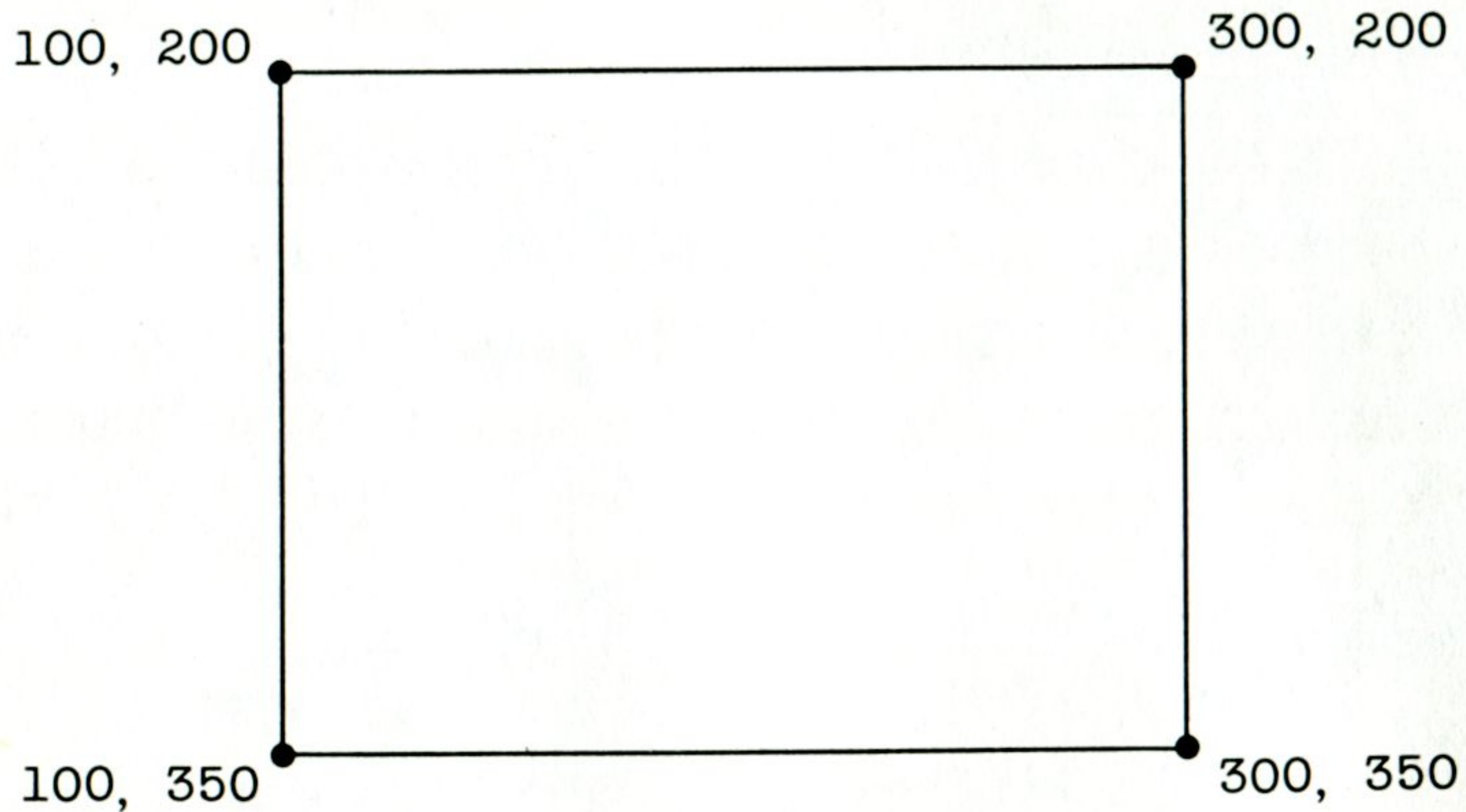
次のプログラムを入力して保存した後，実行してみてください．

LINE．BASプログラム

```
SCREEN 88,  , 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
LINE (100, 200)-(300, 200), 5
END
```

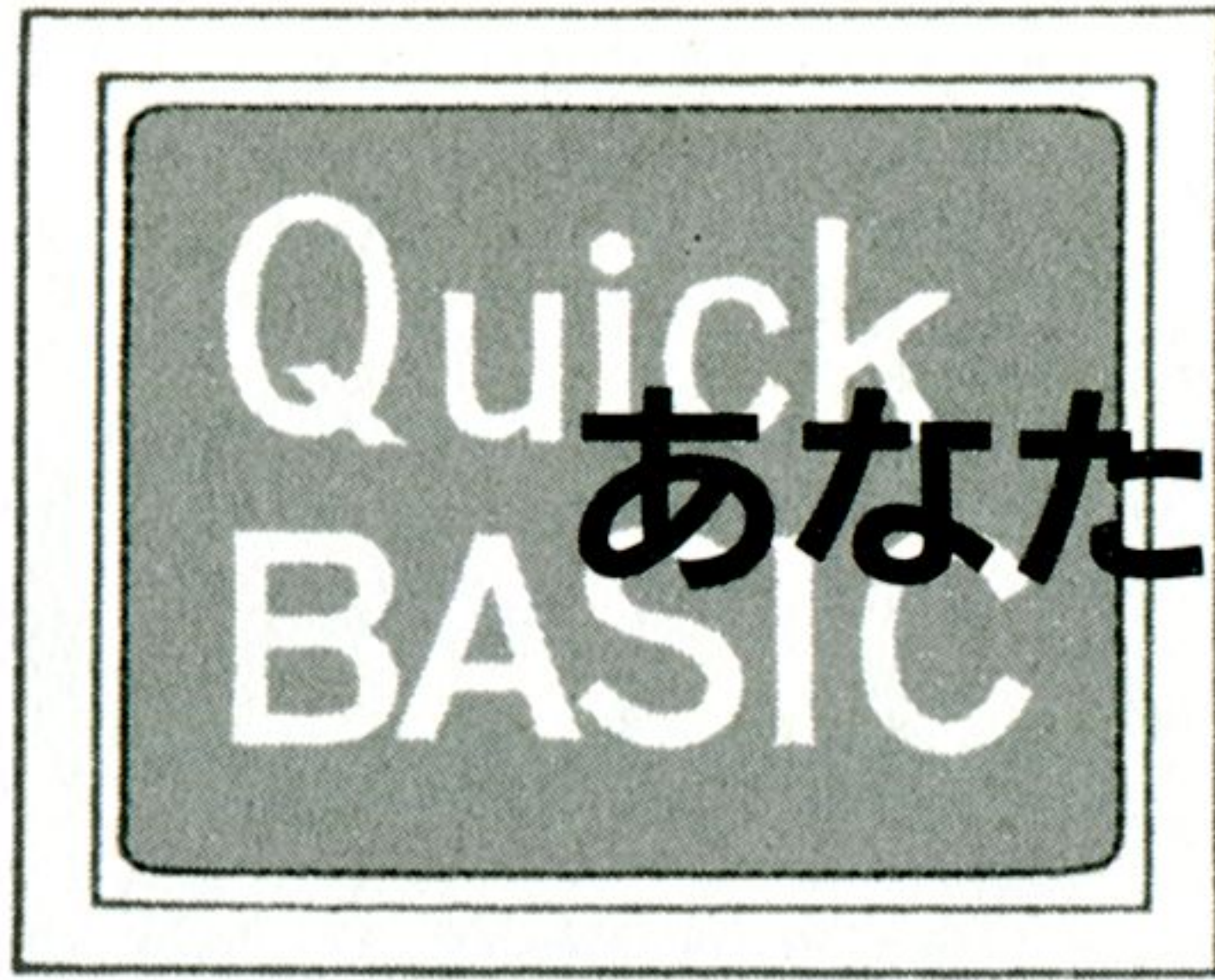
LINE_(開始点の横方向の位置,_開始点の縦方向の位置)－(終点の横方向の位置,_終点の縦方向の位置),_色

LINE(ライン)命令のあとの数字は,上に書かれているような意味を持ちます.この数字をいろいろと変えて試してみてください.

100, 200　　300, 200

100, 350　　300, 350

ここで,上の図のような四角形をかくプログラムを考えてください.どういうプログラムになりましたか? 次に1つの見本を示しておきます.形さえきちんとしていれば,別にどの線を先にかくかはあなたの自由です.

```
LINEBOXA.BASプログラム
  SCREEN 88, , 1, 1
  WINDOW SCREEN (0, 0)-(639, 399)
  CLS
  LINE (100, 200)-(300, 200), 5
  LINE (300, 200)-(300, 350), 5
  LINE (300, 350)-(100, 350), 5
  LINE (100, 350)-(100, 200), 5
  END
```

Quick BASIC

あなたは才能があります

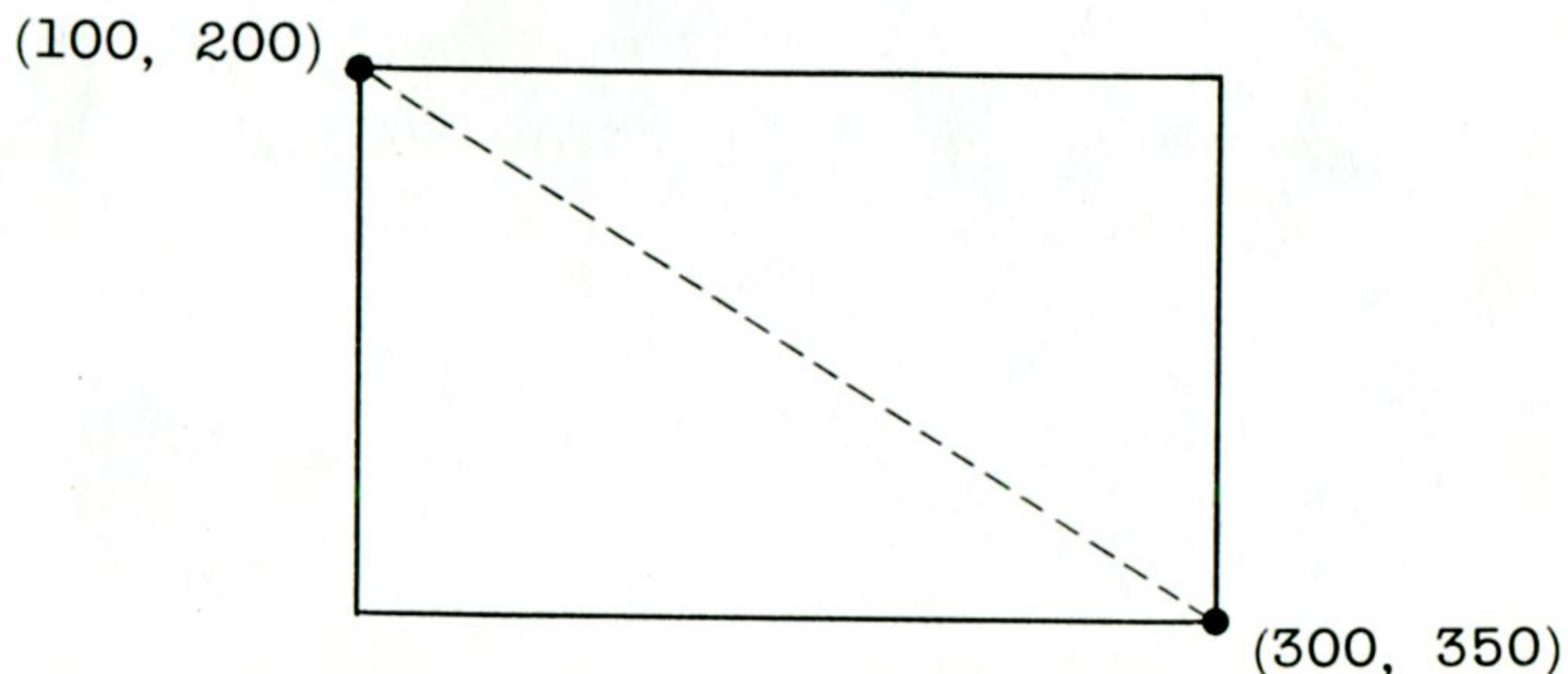
前ページのプログラムで，同じようなことを４回もかくのは面倒くさいと思ったあなた，あなたには才能があります．そもそも人間の作った道具のほとんどが，もっと楽をしたいとか，もっと良い暮らしがしたいという欲求からできたものなのです．コンピュータだって，面倒な計算は機械(マシン)に押し付けようとして，考えられたものなのですから….

次のプログラムを入力して実行してみてください．

```
ＬＩＮＥＢＯＸＢ．ＢＡＳプログラム
 SCREEN 88, , 1, 1
 WINDOW SCREEN (0, 0)-(639, 399)
 CLS
 LINE (100, 200)-(300, 350), 5, B
 END
```

先程の LINE 文と，ほんの少ししか違わないのに，１行で箱がかけてしまいました．

LINE_(100,_200)-(300,_350),_5,_B

上記のように，色のあとに，B を付けると，上記２点を対角線とした箱をかいてくれます．

(100, 200)

(300, 350)

ここで Bの代わりにBFとして入力し直し，実行してみてください．

Bの代わりにBFと入力すると，線だけで四角をかく代わりに，線の中を同じ色で塗りつぶしてくれます．

LINESTYL．BASプログラム

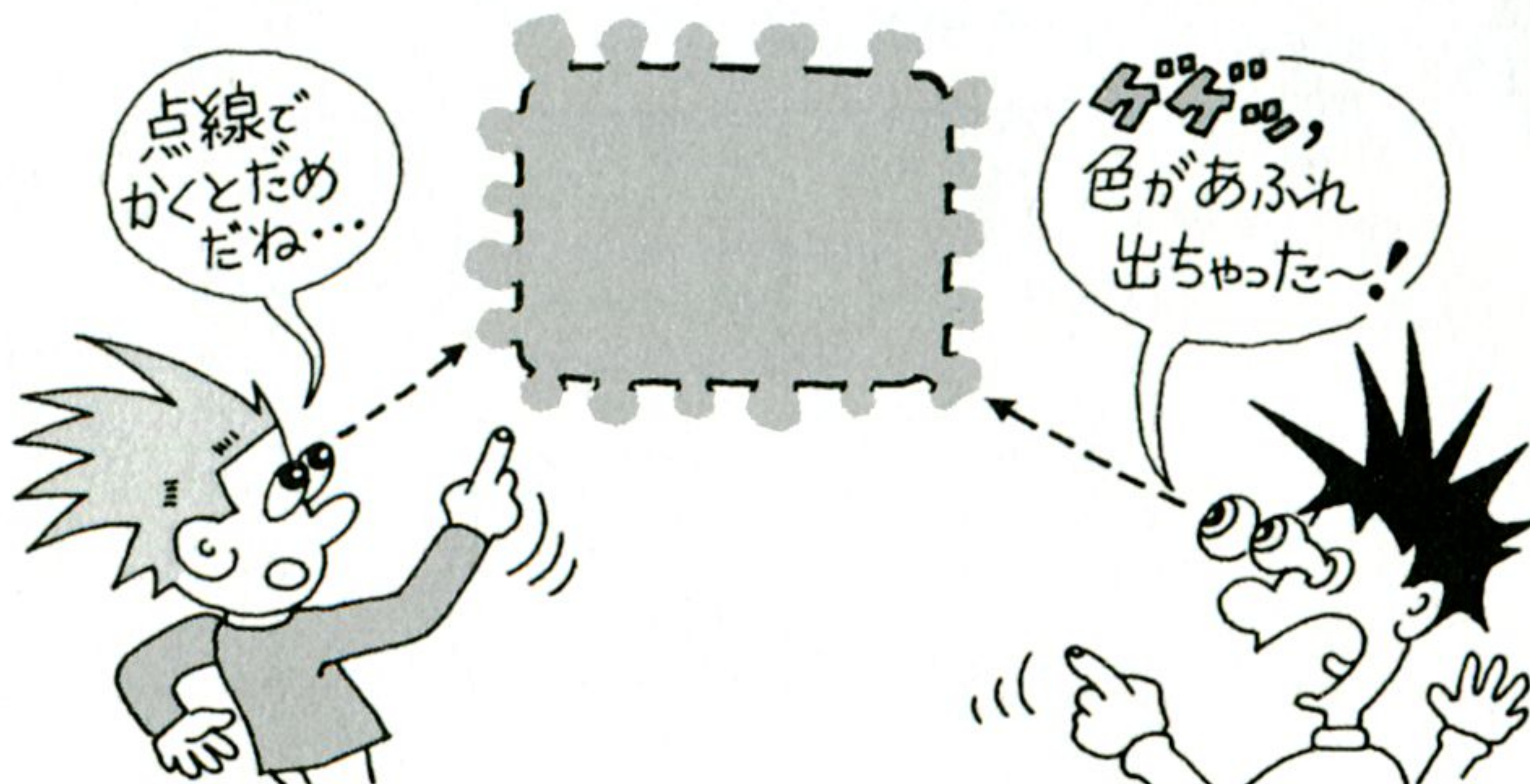
```
SCREEN 88, , 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
LINE (100, 200)-(300, 350), 5, B, &H1010
END
```

では，上のプログラムを入力して実行してください．これには，先程保存しておいたプログラムLINEBOX.BASを呼び出して修正して使いましょう．一番最初のプログラムとほとんど同じです．違うところはBのあとに ,_&H1010などという記号が付いているところです．

このプログラムを実行すると，線が点線になって四角がかかれます．

&Hは，そのあとに来る数が16進数だということを示しています．そのあとの数字は，0からFまでの値のどれかを4個書きます．そして，&HFFFFにすると，普通の直線になってしまいます．他の数字に替えると，その数字に対応した線になります．いろいろな数字に替えて試してみてください．

16進数の数字は0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，Fの16種類の文字で表します．

四角を点線でかいた場合，BFで中を塗りつぶすことはできません．また，点線でかいた四角の中をあとでPAINT(ペイント)命令で色を塗ると，画面全体に色がはみ出してしまいます．御用心！　御用心！

Quick BASIC

ペンキ屋さん (PALETTE,PAINT)

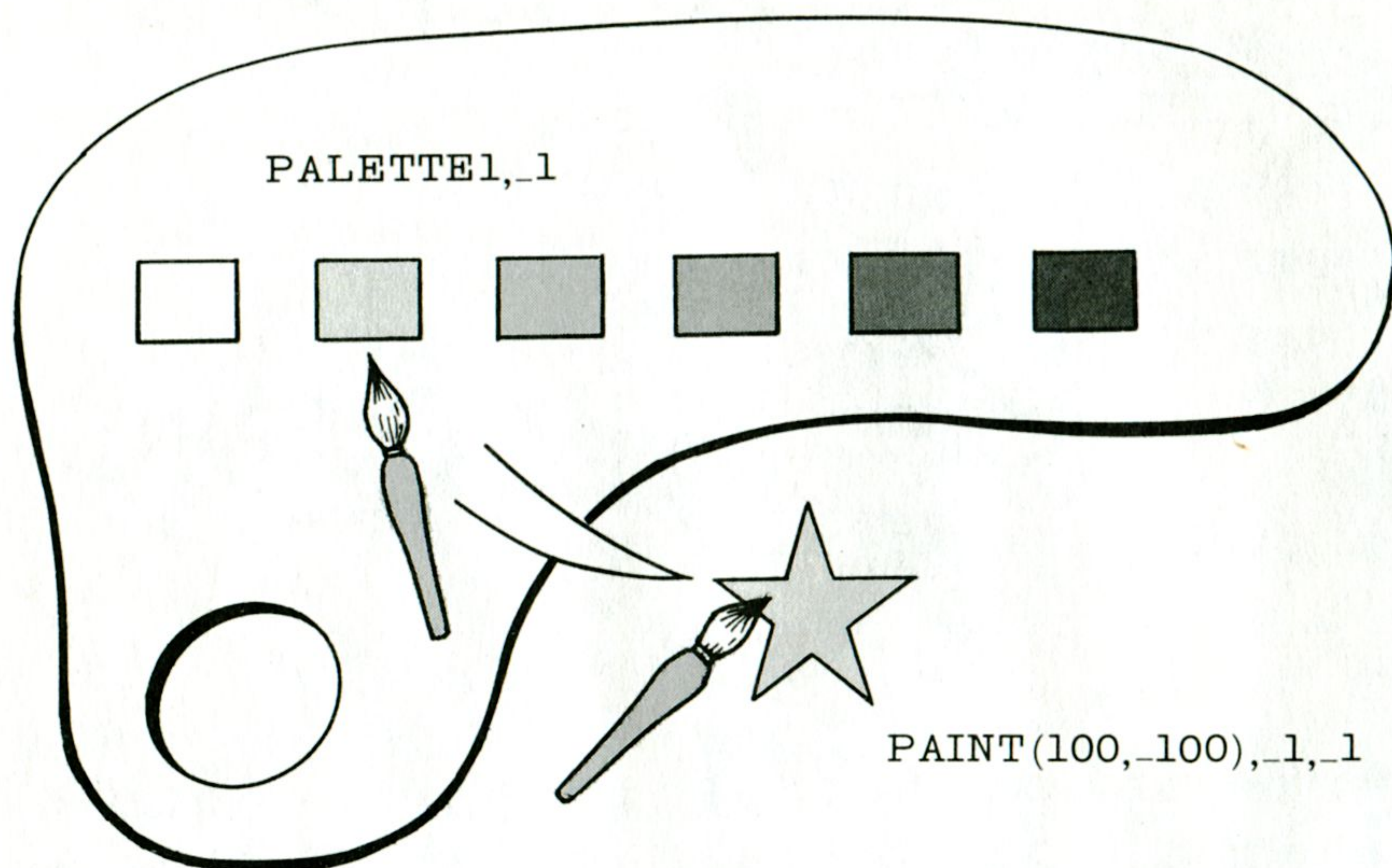
PAINT

サンフランシスコの街を歩いていると，よくビルの壁にいろいろな絵がかいてあります．普通，シスコの建物は隣とピッタリくっついて建てられるのですが，たまたま隣が空地だったりすると，その面一面をキャンパスにしてしまっています．絵は，ビートルズから風景画まで，もうこれは完全なアートです．

私達のプログラムも，ただ線をかいているだけではつまりません．ぜひ色を塗ってやることにしましょう．

BASICでは，色を塗るのにパレットという概念を使っています．これは，まず各パレットの色を決めます．次に色を塗るときに，どのパレットの色を塗るかということを決めて行きます．

PALETTE1,_1

PAINT(100,_100),_1,_1

直接色を塗れば良いのに何でこんな面倒なことをするかというと，それにはワケがあります．

普通の絵をかくのと違って，パソコンの絵には，おもしろい特性があります．つまり，絵をかいてしまったあとにパレットの色を変えると，それまでかかれていた絵の色も変わってしまうという特性です．

この特性は，ゲームやCG(コンピュータグラフィックス)で長所をはっきします．ゲームで夜のシーンになると，周りの色が暗い色になるものがあります．こういうときに，なにも1つ1つの絵を全部塗り替える必要がなくなります．たとえば，単にパレットの色を暗い色に変えれば良いだけになり，プログラムがとても簡単になると同時に，メモリも大幅に節約できます．もちろん，CGで服の色を替えるのも簡単です．

パレットの色を替える命令PALETTE（パレット）の使い方は，次のとおりです．

PALETTE_パレットの番号,_色

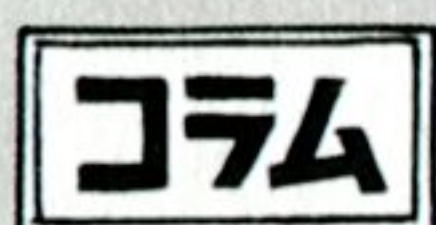
通常，16色中16色モードでは，何も指定しなければパレット番号と色番号は一致しています．

PC-9801VX21，RX，RA，DA…などの機種では，4096色の中から16色を選んで，各パレットに割り当てることができます．この場合，SCREEN 文の最初で，　SCREEN_88,_3　としてください．

各パレットに色を割り当てるときには，色番号は0～4095までの色を選んでください．この数字は次の式で決めます．

色番号＝赤の明るさ×256＋緑の明るさ×16＋青の明るさ

赤，緑，青の明るさは，0から15までの数字を取ることができます．

コラム

色の違い

PC-9801，8801シリーズに使われているN88-BASICとQuick BASICの違いの1つに，色の違いがあります．各パレットに対応する色は，次のようになっています．

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
N88-BASIC	黒	青	赤	紫	緑	水	黄	白	灰	あい	暗赤	暗紫	暗緑	暗水	暗黄	暗白
Quick BASIC	黒	青	緑	水	赤	紫	黄	白	灰	あい	暗緑	暗水	暗赤	暗紫	暗黄	暗白

N88-BASICのプログラムをQuick BASICで実行し，色が問題となる場合，色が一緒になるように，以下のプログラムを最初の方に入れてください．

IROCHNG．BASプログラム

```
SCREEN 88, 1, 1, 1
FOR I = 0 TO 15
PALETTE I, I
NEXT I
PALETTE 2, 4
PALETTE 3, 5
PALETTE 4, 2
PALETTE 5, 3
PALETTE 10, 12
PALETTE 11, 13
PALETTE 12, 10
PALETTE 13, 11
```

このプログラムは16色中16色のモード対応です(90ページのコラム参照)．

色を直接塗る命令 PAINT の使い方は，次のとおりです．

PAINT_(横方向の座標,_縦方向の座標),_パレットの番号,_境界線の色

では，さっそく次のプログラムを入力，保存，実行してみましょう．

ＰＡＩＮＴ．ＢＡＳプログラム

```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
LINE (100, 200)-(300, 350), 5, B
PAINT (150, 250), 3, 5
END
```

どうです．きれいに塗れましたか？

では今度は，同じ四角を青色に塗ってみてください．

PAINT（ペイント）命令を使うときに，とくに注意しなければいけないのは，周りの境界線に切れ目がないようにすることです．少しでも切れ目があると，そこから色が外に流れ出し，画面全部がその色になってしまいます．

とくに下のイラストのように，２つの線のつぎ目のところがしっかりつながっているか，およびかいた線を他の色の部分が切ってすき間があいていないかに注意しましょう．

Quick BASIC

世の中四角より丸がいい？（CIRCLE）

昔から，丸いのは美徳と申しまして，「人間年相応に丸くならなくては」などといわれます．日本では，とくに人格が丸いことを重視します．そういえば，お金の数え方も日本では円と丸の意味です．私は同じ丸でもこの円の方が大好きです．エッ，あなたもですか….

今度は，丸を描く命令CIRCLE（サークル）についてです．
次のプログラムを入力，保存し，実行してみてください．

MARU．BASプログラム
```
SCREEN 88, , 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
CIRCLE (100, 100), 50, 3
END
```

CIRCLE 命令は，次の意味を持ちます．

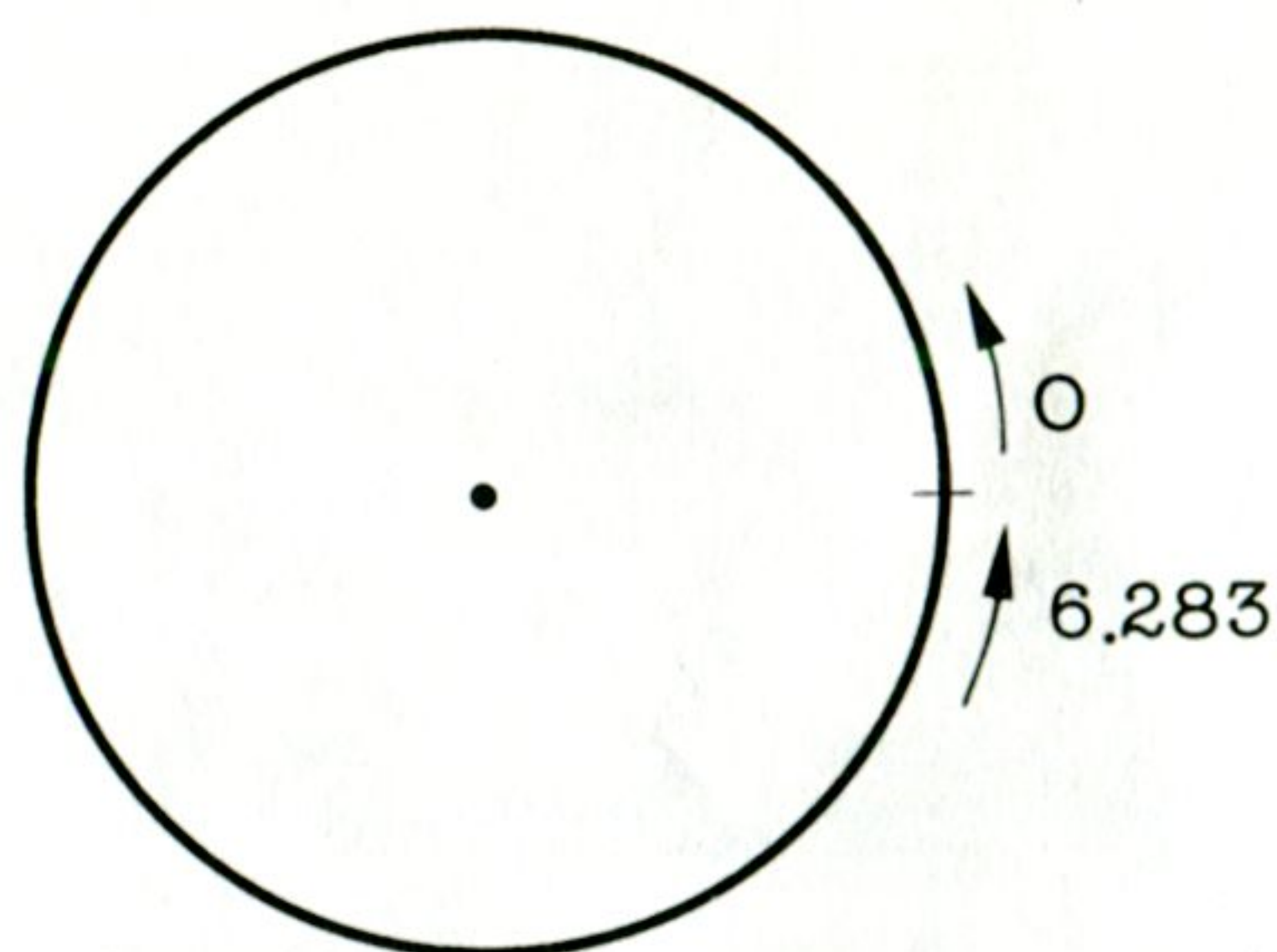
CIRCLE_(横方向の位置,_縦方向の位置),_半径,_色,_開始点,_終了点,縦横比

単に円を描くだけでしたら，色までプログラムをかいて，あとは省略してもかまいません．

ここで，開始点と終了点というのは円をかくときに，どこからかき始めて，どこでかき終えるかということを示しています．

注意しなくてはいけないのは，この値は通常私達が使っている角度ではなくて，ラジアンを使っている点です．ラジアンというのは，円周率 $\pi=3.1415\cdots$ を使って角度を表すやり方です．

つまり，右の図のように，円のかき始めを0として，かき終わりを2π，つまり，6.283…で1周します．別のいい方をすると，半径を1にしたときの円の円周方向の長さを使って角度を表す方法です．ラジアンはわかりにくいから，通常の角度の単位の度を使ったkakuAを使いたいという人は，開始点，終点に使う数を，

0
6.283

ラジアンを使うと

kakuA ＊ 3.1415/180

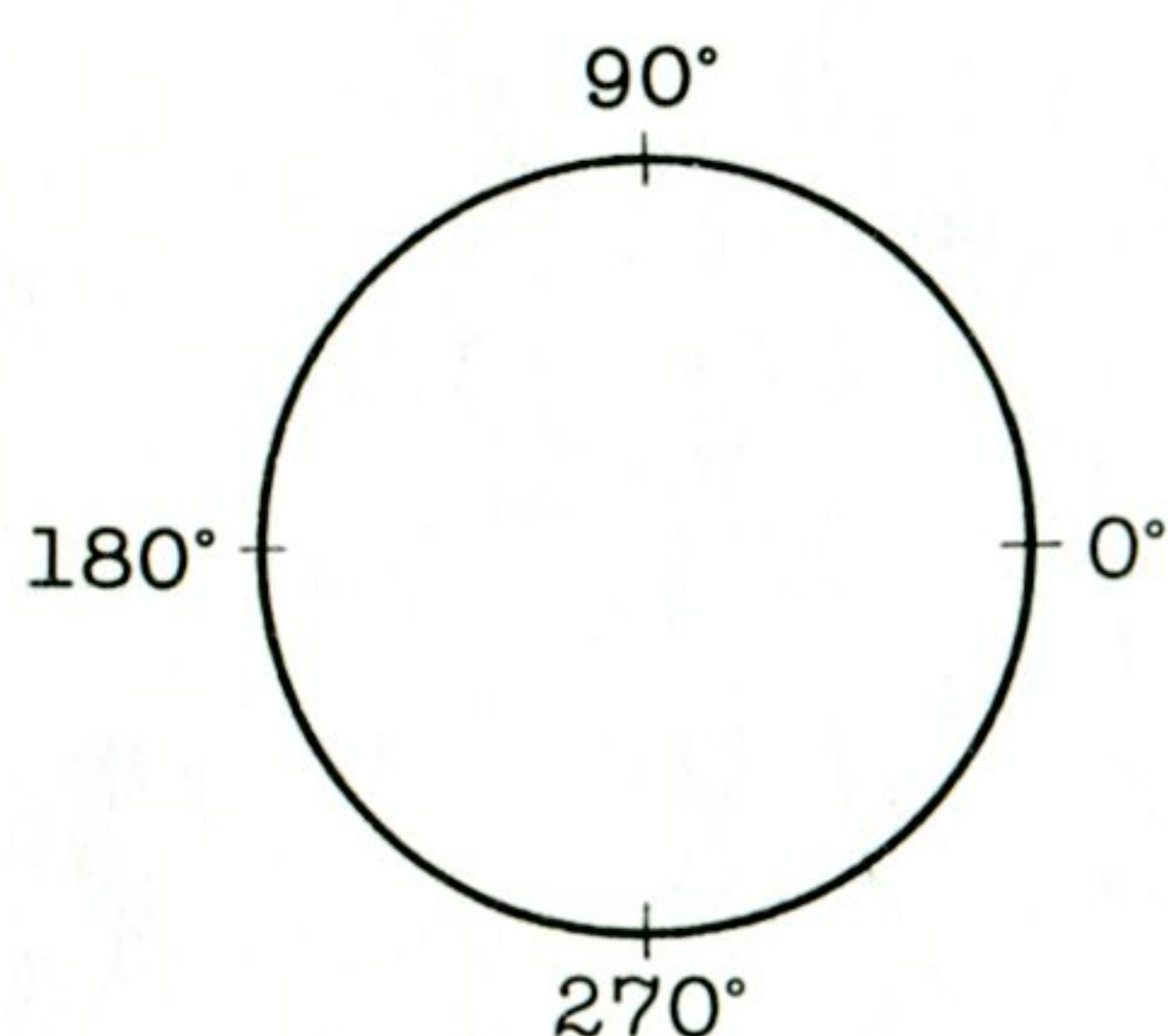
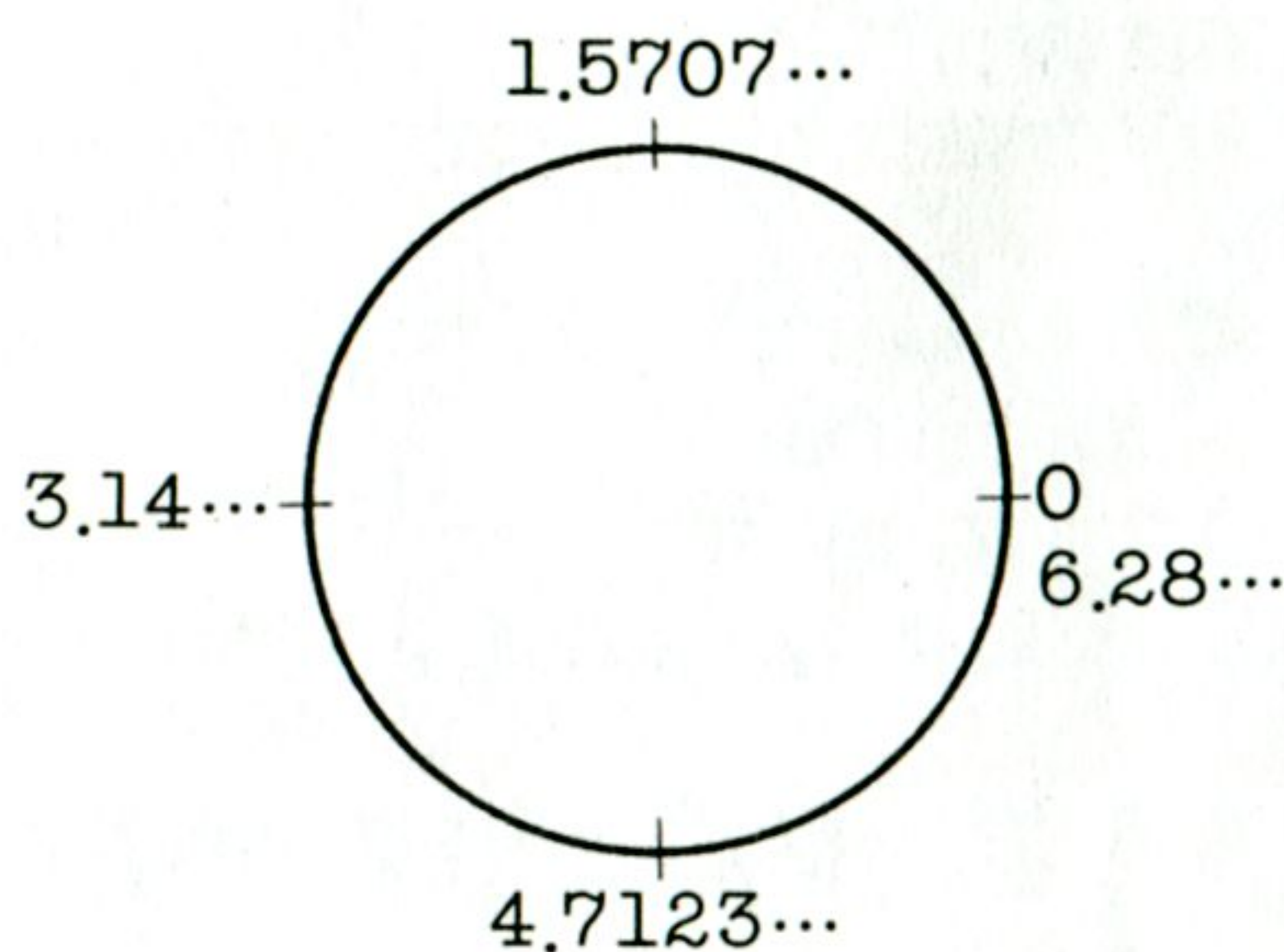
または，

kakuA ＊ 0.017453

のようにすると，kaku Aの値を，いつも私達の使っている度で表すことができます．

90°
180°
0°
270°

度で表した円

1.5707…
3.14…
0
6.28…
4.7123…

ラジアンで表した円

開始点をラジアンで表すと，0のところが，画面の右側に来るようにして円をかき始めます．

また，開始点，終了点の数字の前に－(マイナス)を付けると，円の部分をかくだけでなく，中心からそこまでの線も一緒に描いてくれます．円グラフを作りたいときには，とても便利です．ただし，この－(マイナス)方向を表すわけではないので，逆回りにかくようなことはしません．

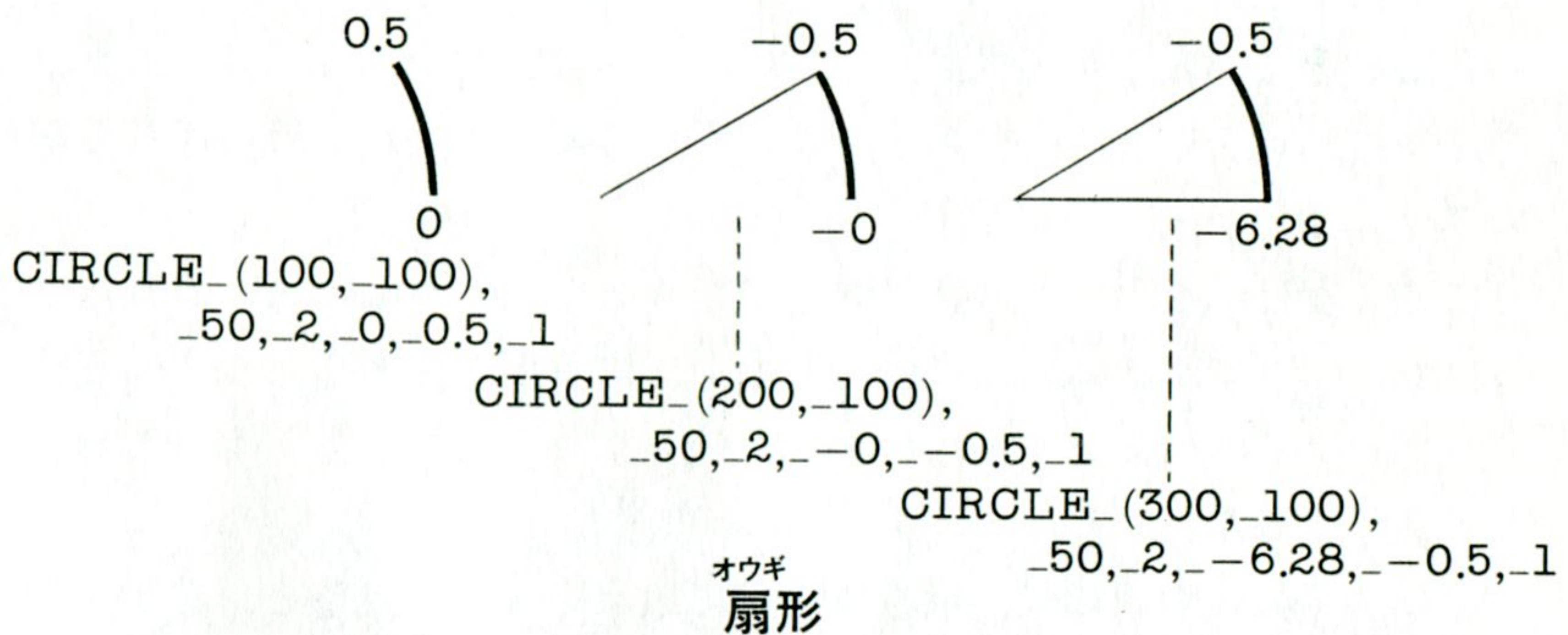
また，－0では，中心からの線をかいてくれないので，－6.28…で代用します．

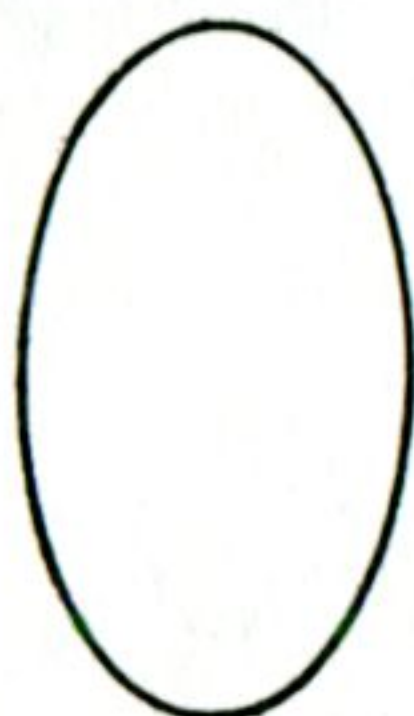
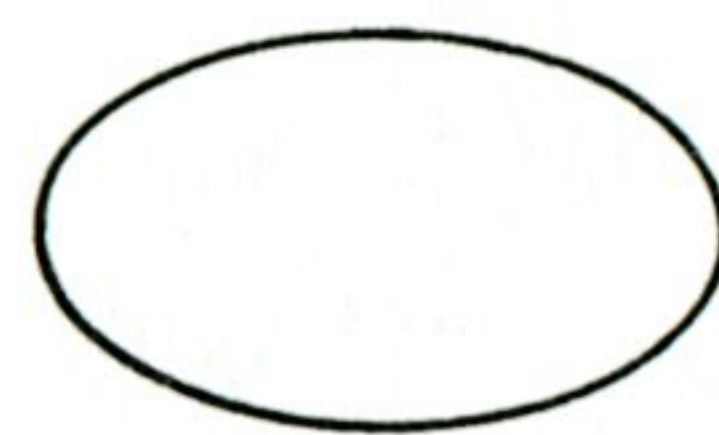
```
MARU2．BAS
  SCREEN 88,  , 1, 1
  WINDOW SCREEN (0, 0)-(639, 399)
  CLS
  CIRCLE (100, 100), 50, 2, 0, .5, 1
  CIRCLE (200, 100), 50, 2, -0, -.5, 1
  CIRCLE (300, 100), 50, 2, -6.28, -.5, 1
  END
```

0.5
0
CIRCLE␣(100,␣100),
␣50,␣2,␣0,␣0.5,␣1

−0.5
−0
CIRCLE␣(200,␣100),
␣50,␣2,␣−0,␣−0.5,␣1

−0.5
−6.28
CIRCLE␣(300,␣100),
␣50,␣2,␣−6.28,␣−0.5,␣1

扇形（オウギ）

最後の縦横比は，円ではなく，だ円をかくときに使います．値が１のときが真円で，１より小さくなると横長のだ円になりますし，１より大きければ縦長のだ円になります．

```
MARU3．BASプログラム
  SCREEN 88, 0, 1, 1
  WINDOW SCREEN (0, 0)-(639, 399)
  CLS
  CIRCLE (200, 200), 50, 5, 0, 6.28, 1.5
  CIRCLE (400, 200), 50, 6, 0, 6.28, .5
  END
```

CIRCLE␣(200,␣200),
␣50,␣5,␣0,␣6.28,␣1.5

CIRCLE␣(400,␣200),
␣50,␣6,␣0,␣6.28,␣0.5

Quick BASIC

態度のころころ変わるやつ，変数？

とある放課後の会話，「小泉今日子てかわいいよなー」，「宮沢リエよりずっといいよな」そこへ先輩が現れて，「宮沢リエ最高だよな」と，「本当，小泉今日子なんか目じゃないっすよ」……どこにもいるんですよね．ほら部長の前だけ巨人ファンとか…．

そこで今度は，パソコンの世界の中で態度の変わるやつ“変数”のお話です．

パソコンのプログラムの中で，変数は非常に重要な働きをします．実際，すべてのプログラムは，ある変数の特別な値を求めるものと極言することもできなくはありません．

パソコンで扱う変数には，大きく分けて数字を扱う数値型と，文字を表す文字型があります．

さらに，数字を表す変数の型には，整数型と小数点型に分れます．

整数型は，小数点以下が表せない数字です．

細かい計算は苦手ですが，値がきっちり出るために，お金の計算などに使います．また，メモリが少なくて済むのも特徴です．

小数点型は，小数点以下が表せる数字です．

細かい計算が得意で，技術計算などに向きます．

Quick BASICの世界では，“整数型”，“長整数型”，“単精度浮動小数点型”，“倍精度浮動小数点型”の4種類の型の数字を表す変数と，“可変長文字列型”，“固定長文字列型”の2種類の文字列型の変数があります．

数値の変数は，とくに意識せずに型名を省略したときには，自動的に単精度の浮動小数点型となります．

各数値変数の使える範囲は，

整 数 型：－32,768から32,767までの整数値

長整数型：－2,147,483,648から2,147,483,647までの整数値

単精度浮動小数点整数の範囲は，

負の値：－3.402823E＋38から－1.175494E－38まで

正の値：＋1.175494E－38から＋3.402823E＋38まで

倍精度浮動小数点型の範囲は，

負の値：－1.797693134862316D＋308から－2.225073858507201D－308まで

正の値：＋2.225073858507201D－308から＋1.797693134862316D＋308まで

と，これだけの数が扱えます．通常，私達が使う数字なら，長整数型と倍精度の浮動小数点まででで十分にカバーされています．

とくにある変数を，ある型にしたい場合は，変数を使う前にその変数の型を宣言します．

宣言は，次のようにします．

整数型	DEFINT（デファインインテジャー） mouke
長整数型	DEFLNG（デファインロング） chokin
単精度浮動小数点	DEFSNG（デファインシングル） nenpi
倍精度浮動小数点	DEFDBL（デファインダブル） heikin

また，次のように書くこともできます．

整数型	mouke%＝51230
長整数型	chokin&＝213726371
単精度浮動小数点	nenpi!＝5.763897E＋12
倍精度浮動小数点	heikin#＝2.21035108921316D＋28

％＆！＃　は，変数名のあとに付けて，その変数がどういう型なのかを示します．

E＋12は，大きな数を表すときに使い，その前の数に10の12乗(10^{12})を掛けた値であることを示します．

E－12とマイナスが付いた場合，非常に小さな数を表すのに使い，その前の数を10の12乗(10^{12})で割った数であることを示します．

DもEと同じ意味ですが，倍精度の浮動小数点に使います．

DIM（ディム）(ディメンション)を使って，次のように宣言することもできます．

```
DIM mouke AS INTEGER
DIM chokin AS LONG
DIM nenpi AS SINGLE
DIM heikin AS DOUBLE
```

Quick BASIC

コンニチハ赤ちゃん（変数のネーミング）

上のような名前の歌がありました．自分の子供が生まれるというのは不思議な気がします．少なくとも半分は自分と同じ遺伝子を持った人間がこの世の中に出て来るのですから．

子供が生まれて，最初に悩むのが，どういう名前を付けるかです．なにせ一生つきまとって行くのですから，変な名前にしてしまってはかわいそうです．そうです，プログラムの中の変数にも名前が必要なのです．上手な名前を付けてあげましょう．

変数の名前の付け方で一番重要なのは，わかりやすい名前を付けることです．昔は，メモリを節約するために，「Aだ」，「Bだ」と，アルファベット１文字でやっていましたが，これにしてしまうと，あとでなにがなんだかわからなくなります．幸い，Quick BASIC では，40文字までの名前が付けられますので，AKIRA というように，だれが見てもよくわかる名前を付けましょう．

Quick BASIC で変数の名前に許される文字の種類は，アルファベットと数字です．このとき，必ずアルファベットで始めるようにしてください．また，Quick BASIC の中で使っている命令語は，当然，変数名としては使えません．

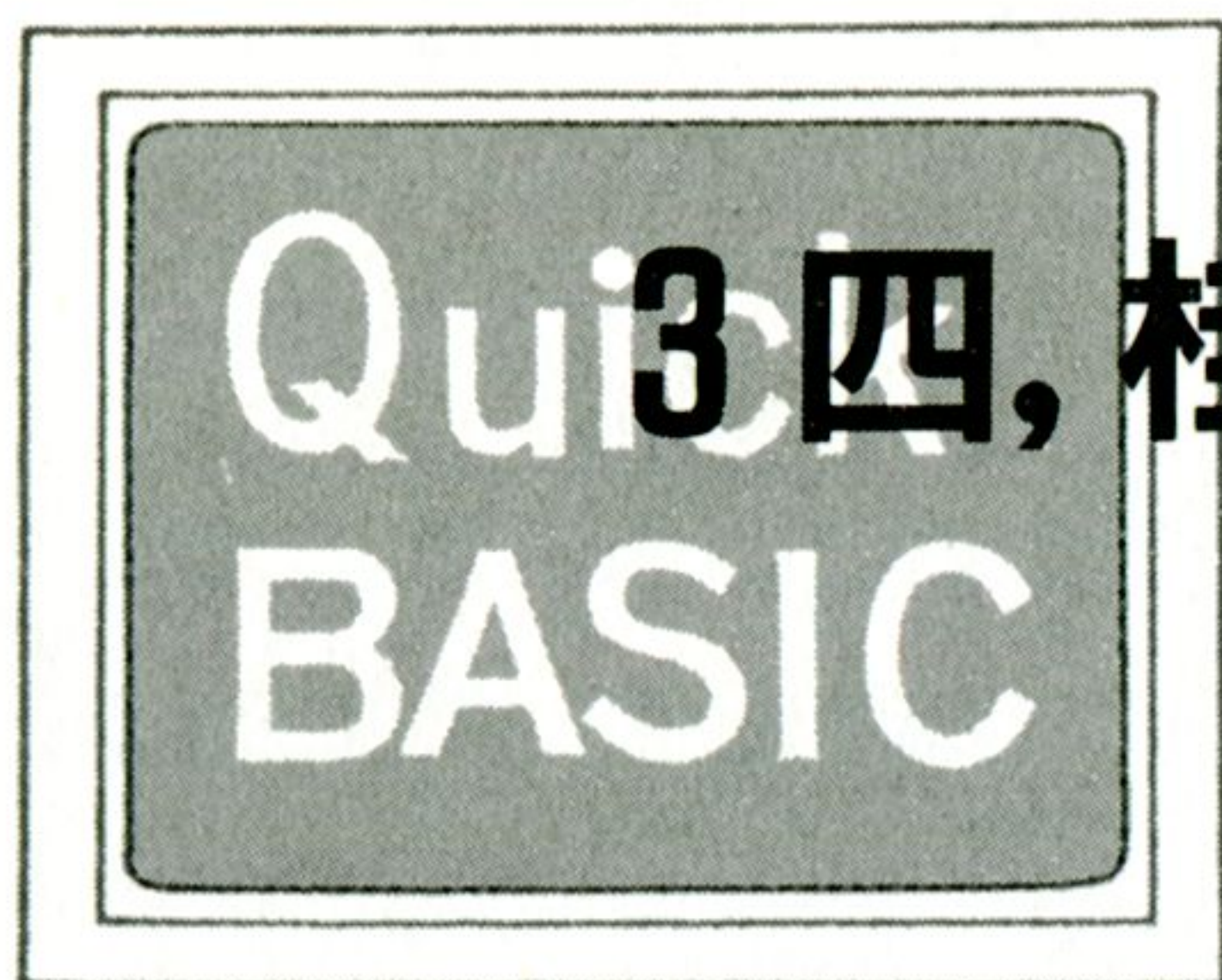
ただし，命令語を名前の一部に含む変数は受け付けてくれます．

× PRINT＝10

○ APRINT＝10

C言語のように，途中をアンダーラインで結んだ変数は受け付けませんので注意してください．

Quick BASIC は，変数の大文字・小文字を区別していません．便利なことに同じスペルでしたら，一番最後に入力したように，すべての同じ変数を書き換えてくれます．

Quick BASIC

3四，桂馬，王手飛車トリ，マッター（配列）

よくある光景ですが，本当のルールではマッタはなしです(もしマッタをしたら，即マケです).

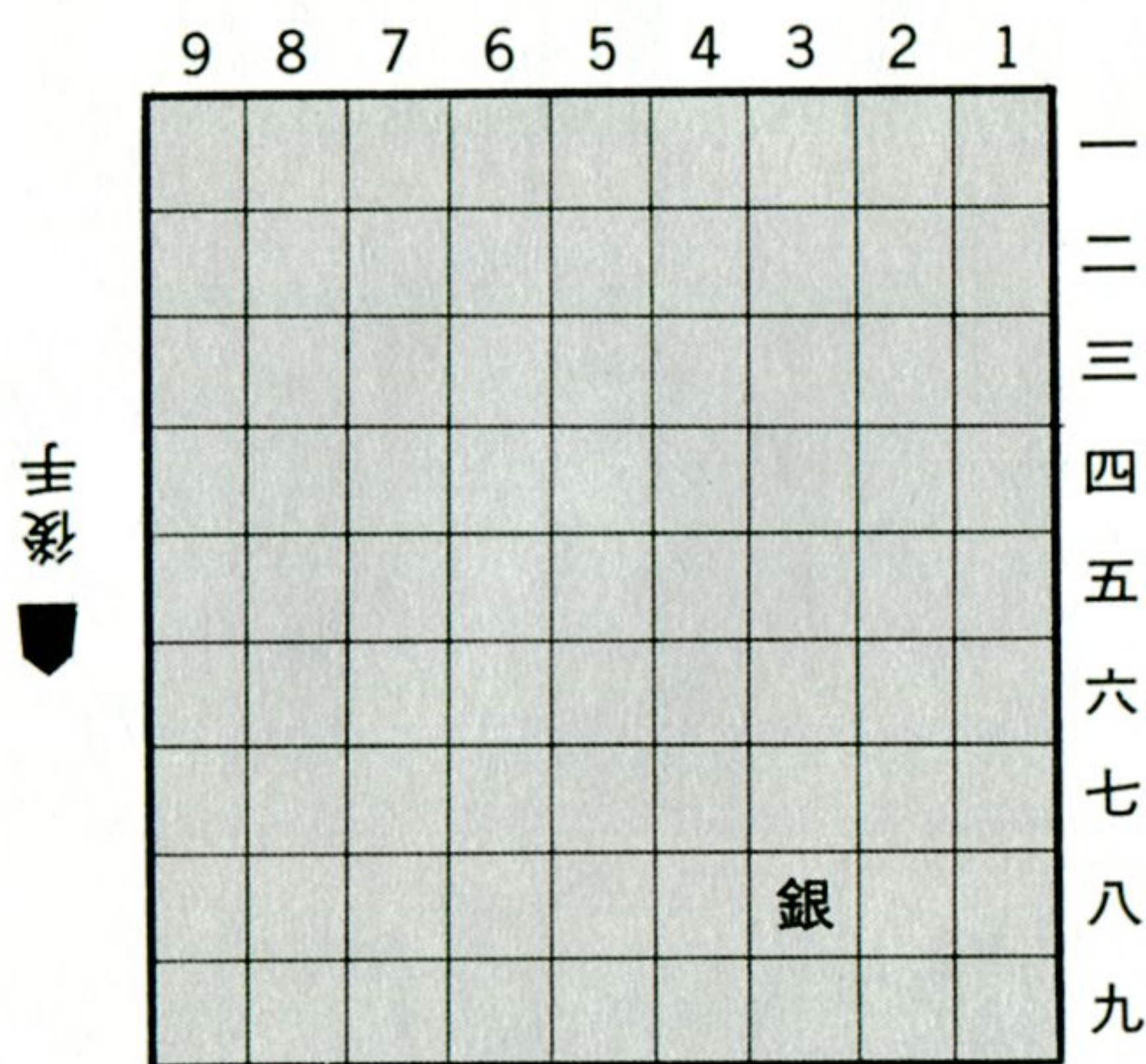
ところで，一番前に書いてある3四とは何なのでしょう.

9 8 7 6 5 4 3 2 1

一 二 三 四 五 六 七 八 九

後手

銀

先手

駒の位置は，
☖3八銀のようにアラビア数字，
漢数字の順で表します.

先手則から見ます.

よくご存知のように，将棋板は9×9のマスでできています．ここで，だれがどの駒をどう進めたかについて，なんらかの決め方がないと，「王の右横2つ目の銀が右斜目前に動いて…」などと，わけがわからなくなってしまいます．そこで，上図のように，アラビア数字と漢数字で分けて場所を簡単に表しています.

もうこれは，立派な配列といえます.

要するに配列とは，ある場所が簡単にわかるように基準を決めた入れものを作り，その中で，どこになにがあるかを示すものだと思ってください.

配列には，単純に順番に番号を増やしていく1次元の配列と，将棋板のように縦，横2つの基準で示す2次元の配列，さらにそれに高さを加えた3次元の配列と，次元をどんどん増やして行くことができます.

HAIRETU．BASプログラム

```
CLS
DIM Y%(80)
Y%(1) = 1
Y%(2) = 2
Y%(3) = 3
X = Y%(1)
PRINT X
END
```

このプログラムに書かれているDIMが配列の宣言です．DIM_Y%(80)　のY%がＹという配列変数をパソコンのメモリ上に取り，その変数は整数型で，80個分確保しなさいという命令です．

%は，整数型だということを示しています．

Ｙがどのような数かということにより，記号が次のように変わります．

長　整　数　　DIM_Y&(80)

単精度実数　　DIM_Y!(80)

倍精度実数　　DIM_Y #(80)

また，次のような書き方もできます．

整　　　数　　DIM_Y(80)_AS_INTEGER

長　整　数　　DIM_Y(80)_AS LONG

単精度実数　　DIM_Y(80)_AS SINGLE

倍精度実数　　DIM_Y(80)_AS DOUBLE

1つ1つが5文字までの文字変数

DIM_Y(80)_AS_STRING_＊5

2次元の配列の場合

DIM_Y%(8,_10)

のように表します．

これは，メモリ上に9列×11行，つまり99個分の整数の領域を確保しなさいという命令です．確保したあとの1つ1つのメモリ部分に何が入るかは，Y%(2,_1)　のように入る場所を書いて表します．

注）配列は0から始まるため，8，10とすると，9×11個分のメモリが確保されます．

次のプログラムを入力して，実行してみてください．

```
HAIRETU2．BASプログラム
  CLS
  DIM Y%(8, 10)
  Y%(2, 1) = 30
  Y%(2, 2) = 33
  X = Y%(2, 1)
  PRINT X
  END
```

将棋板を配列として宣言すると，

DIM_SHOUGI(9,_9)_AS_STRING_＊_10

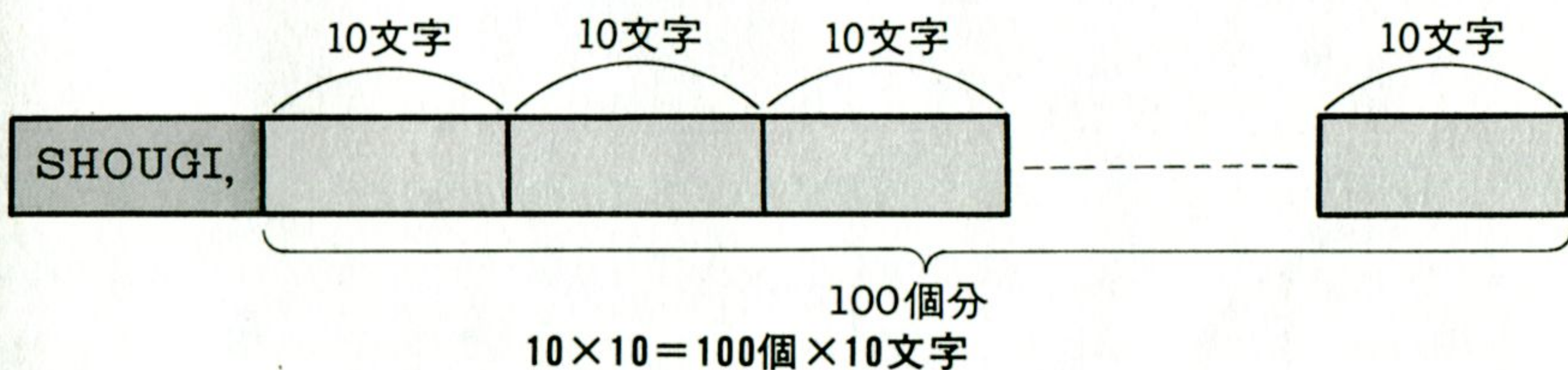
のようになります．

これは，メモリ上に10個×10個＝100個の場所を確保し，そのおのおのの場所は，文字が10個入る分だけ確保しなさいという命令です．

この命令を受けると，パソコンは，次のようにメモリを確保します．

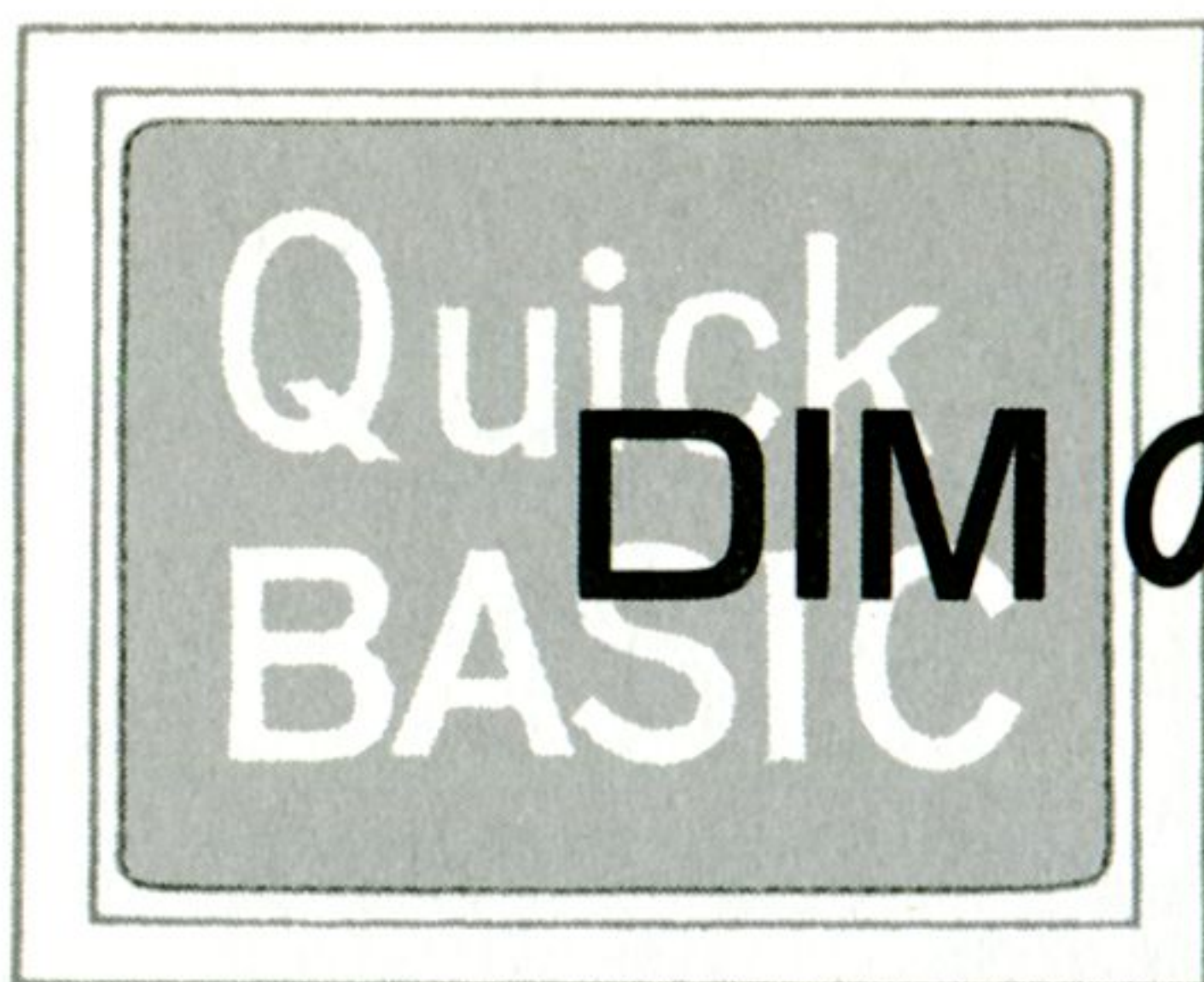
SHOUGI,	10文字	10文字	10文字	----------	10文字

100個分

10×10＝100個×10文字

注）配列は0から始まるため，9，9とすると，10×10の100個分メモリが確保されてしまいます．その中で9×9の81個を利用します．

では，この配列を使ったプログラムを実験してみましょう．

```
SHOUGI．BASプログラム
  CLS
  DIM SHOUGI(9, 9) AS STRING * 10
  SHOUGI(5, 9) = "OU"
  SHOUGI(3, 9) = "GIN"
  PRINT SHOUGI(3, 9)
  END
```

Quick BASIC

DIMのたとえ話

DIM

人間は，3次元の世界に住んでいます．これが6次元だ，7次元だなどというと，わけがわからなくなります．DIMの世界でいう次元とは，これとは少し違うようです．たとえば，あなたが家を作って貸す人だとします．5人ぐらいがワンルームを借りに来たら，5室を平屋でつなげればいいわけです．もっと増えて20人になったら20軒真直ぐにつなぐわけにはいかないので，5室を4階建にすればいいわけです．これでも足りなくなれば，1号棟，2号棟…と増やしていき，さらに必要なら第1団地，第2団地と増やせば良いわけです．

整数とか文字列とかを書くのは，1軒ごとの大きさを決めているわけです．実際には，ワンルームでいいとか，3DKが欲しいとか，いろいろリクエストは違います．(　　)の中の数が，その型の家を何軒作るかを決めています．

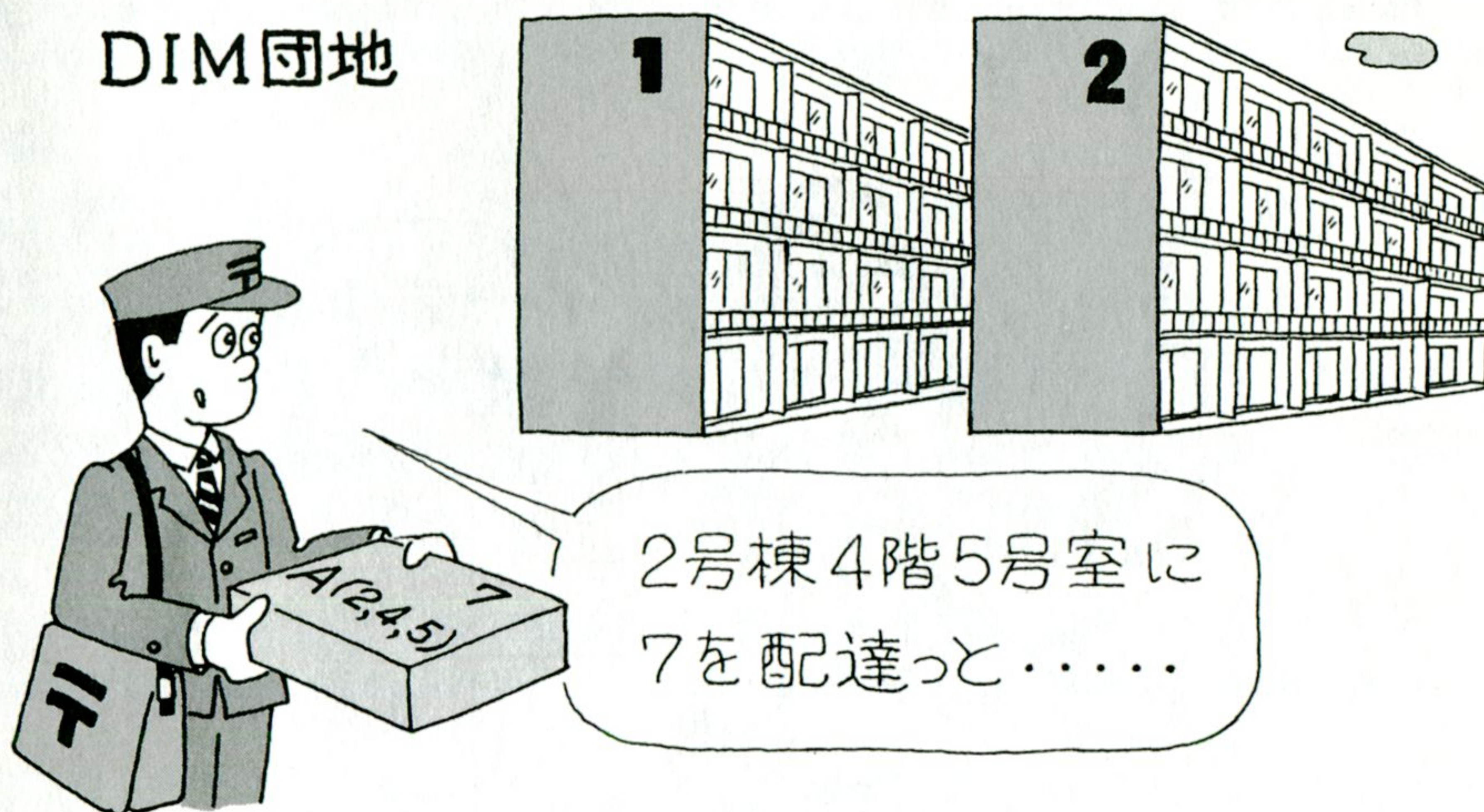
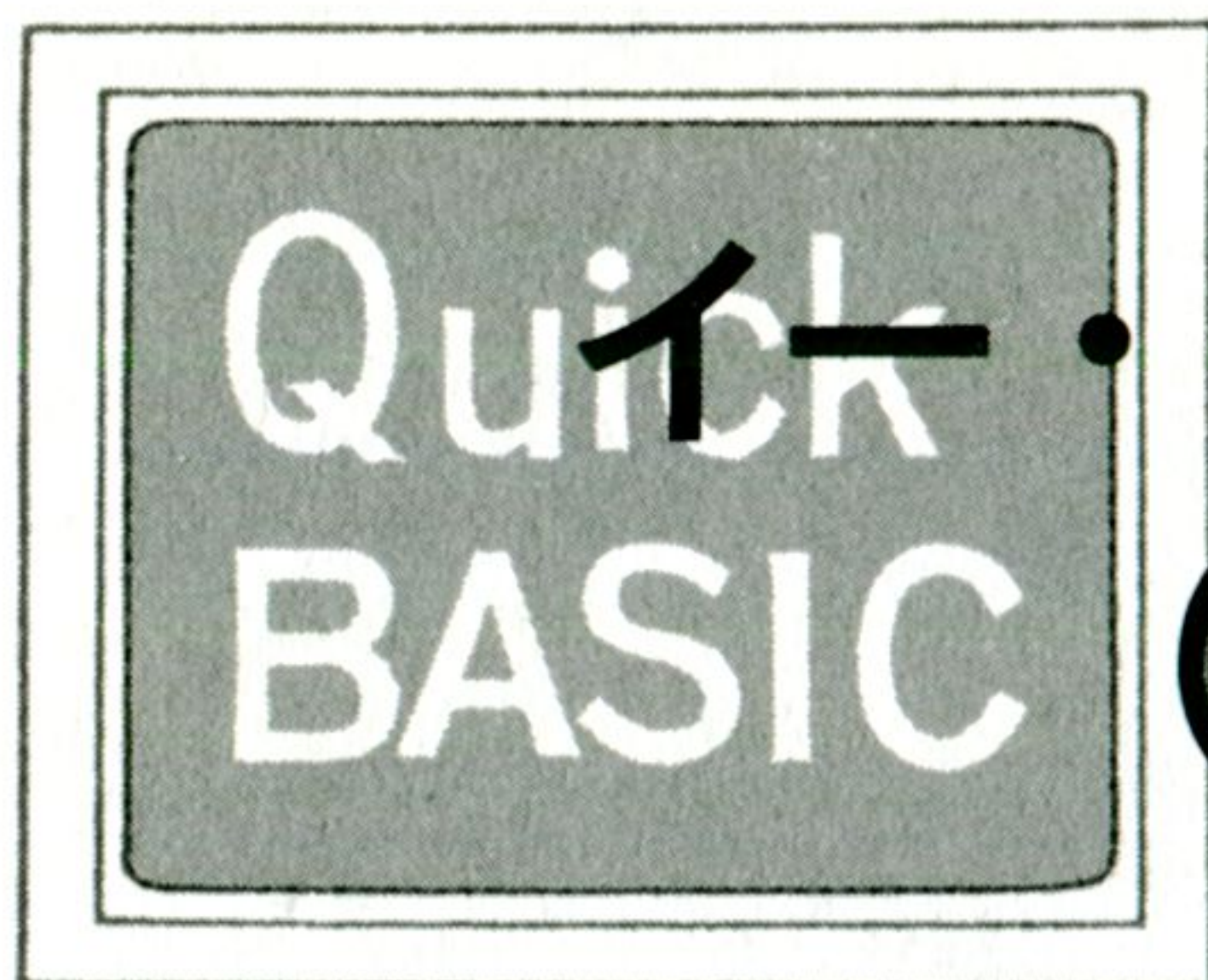
DIMで定義した場所への数や字の入れ方と取り出し方は，次のようになります．

```
DIMINOUT.BASプログラム
  DIM A(5, 5)
  A(1, 1) = 4
  A(1, 2) = 3
  A(5, 1) = 5
  A(5, 2) = 6
  B = A(5, 2)
  PRINT B
```

DIMで定義したメモリ上のある場所に，数を入れるのは簡単です．左辺に，A(5,_2)　のように入れたい場所を(　　)の中に書いた変数をおいておき，＝でつないで，右辺にDIMで定義したときと同じ型の数を書いてやれば良いのです．

あとで入れておいた数が何だったか知りたい時は，Bという変数を作り，＝でつないで変数Bに　A(5,_2)　の値を右辺において代入してやれば良いのです．

B_=_A(5,_2)

Quick BASIC

イー・アル・サン・スー（中国の数の数え方）

突然の中国語で，ビックリさせてしまってゴメンなさい．でも，どことなく日本のいい方と似ていませんか？

そこで，ここでは，サン・スー　ならぬ算数のお話です．

算数というと，だれでも思い浮かべるのが，足し算・引き算・掛け算・割り算です．ここで，BASIC の世界では，次のように表します．

足し算	A＝1＋2
引き算	B＝2−1
掛け算	C＝1＊2
割り算	D＝1／2

足し算・引き算はすんなり納得ですが，掛け算と割り算はちょっと見なれない書き方です．

実は，これはコンピュータが表すことができる文字の問題からきているのです．

コンピュータでは，×という記号を持っていません．似ているとすればXですが，これでは，XXYと書いたときに，XかけるYなのかXXYという変数なのか区別がつかなくなります．そこで，×にもう1本線を入れた＊ならマア似ているから，と作ったのがこの掛け算の記号なのです．

同じように割り算も，1÷2なら$\frac{1}{2}$，これでは1行にかけないので，½となり，さらに文字を分けて1／2となりました．

他の算術記号としては，べき乗に＾を使います．つまり，3^2，3を2乗するは　3＾2　のように書きます．

べき乗	3^2	3＾2
	$4^{2.3}$	4＾2.3
ルート(平方根)	2.0	SQR(2.0)

¥　　整数の割り算で，その商を計算します．つまり，10÷3＝3…1のときに，商の3を返します．

使い方としては，

A_=_10_¥_3

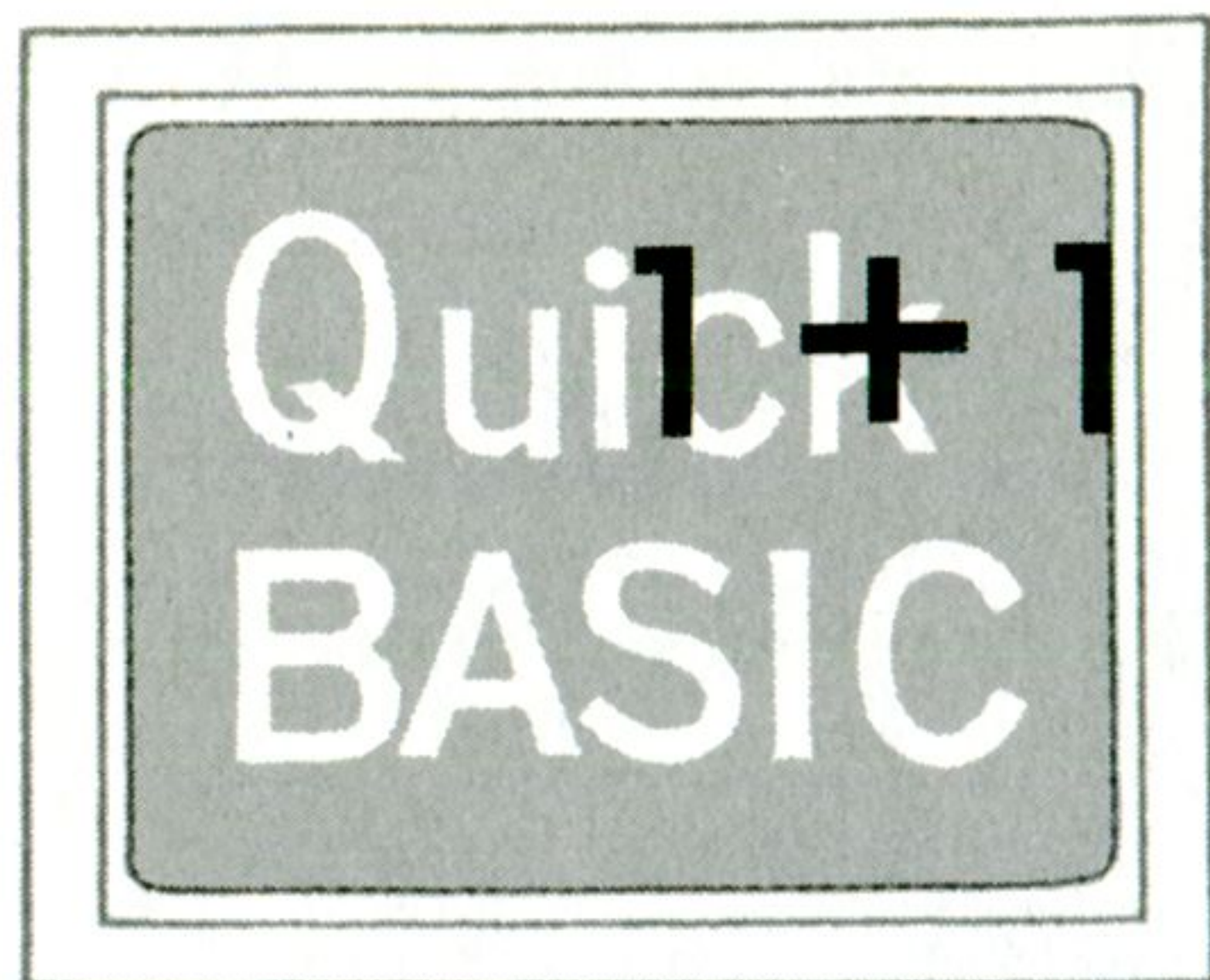
のように使います．

MOD　整数の割り算で，商の部分はまったく無視して余りがいくつになるかを返します．つまり，10÷3＝3…1のときに1を返します．

使い方としては，

A_=_10_MOD_3

のように使います．

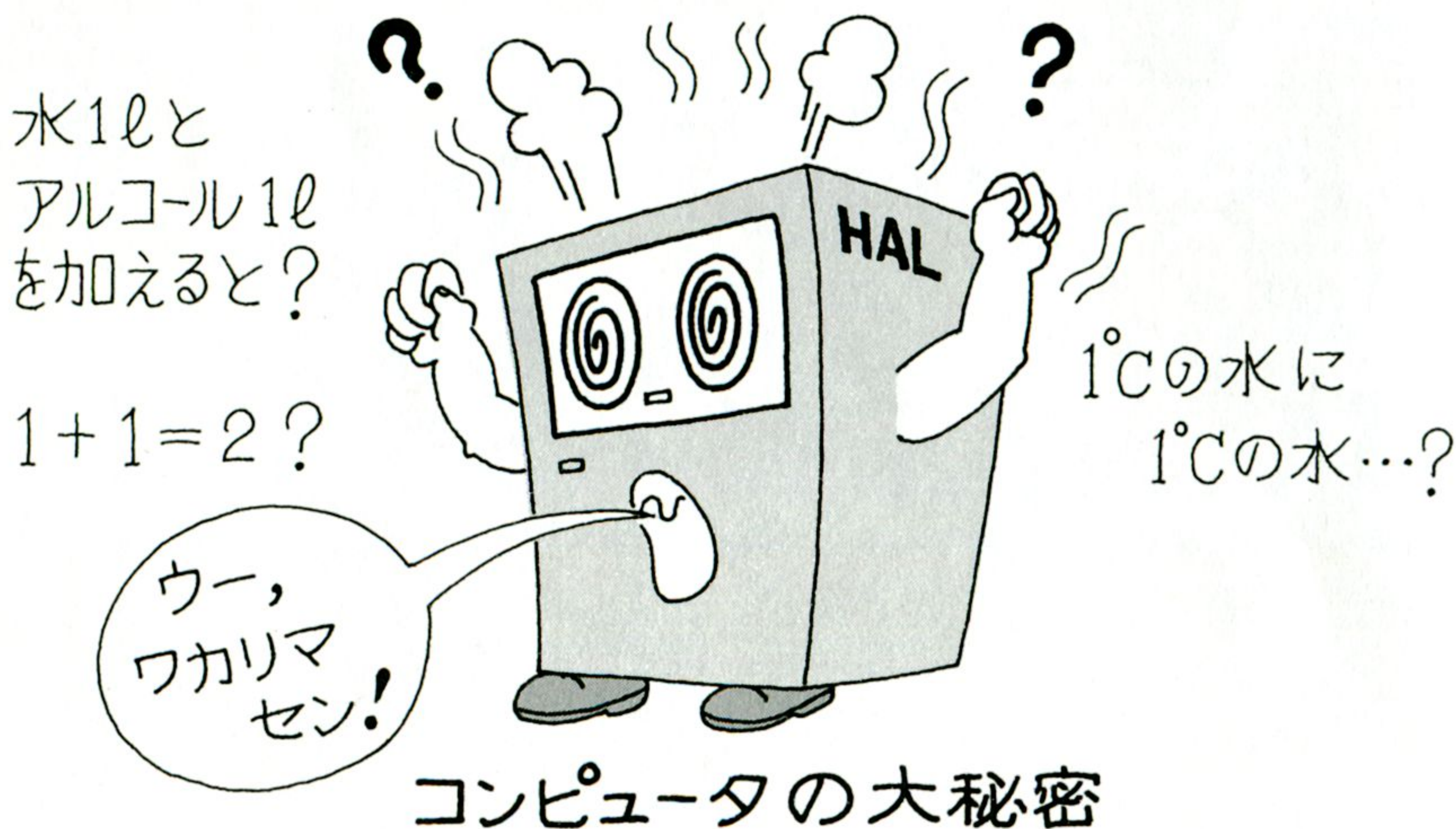
Quick BASIC

1＋1＝2は知りません？（式の作り方）

コンピュータは，もともと計算を速く楽にするために作られた道具ですから，計算するのが得意です．絵をかいたり音を出したりするのも，ある種の計算をしているといえなくもありません．でも，世の中いつも計算どおりにはいきません．たとえば，1＋1はいつも2だと思っているあなたへ次のような質問をしてみます．

(1)　1℃の水1 lに1℃の水1 lを加えたときの水の温度は？
　　(答は2℃ではありません)

(2)　水1 lとアルコール1 l両方加えると何 l？

上の問題は，まあクイズみたいなものですが，コンピュータの扱う式は，いつも私達が使っている式とは少し違います．たとえば，左から右に1＋1＝Aという書き方ではコンピュータは理解してくれません．

コンピュータの大秘密

Quick BASICでは，必ず答が代入されるべき変数が左側に単独でこなくてはなりません．つまり，1＋1の結果をAという変数に入れたければ，

○　A_=_1_+_1

×　1_+_1_=_A

また，方程式のように，

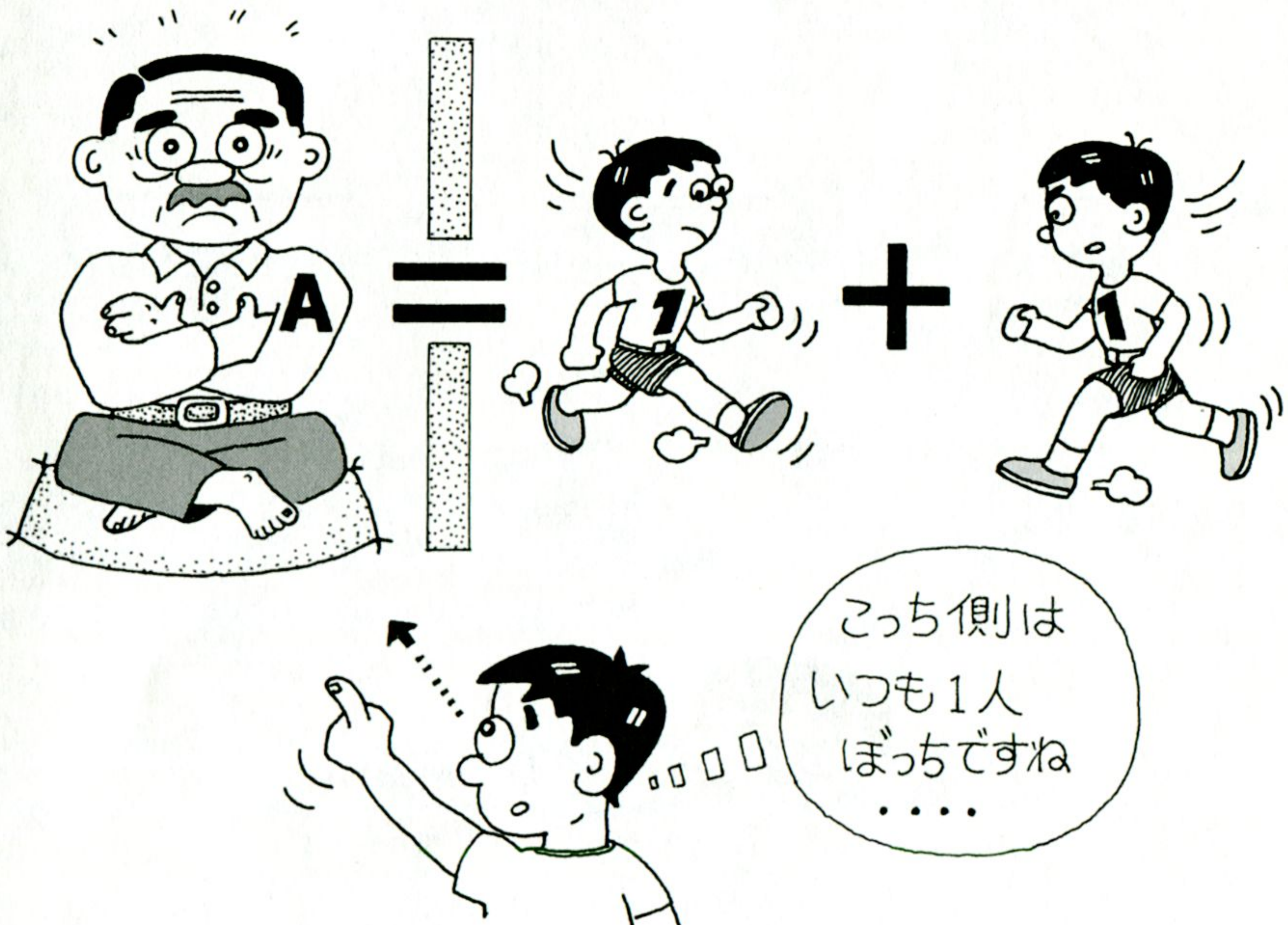
A_+_2_=_6

のような書き方もできません．

自分で書く時には，変数は必ず等合の左辺に，単独で存在するように書く，ということ守ってください．

前ページの答は

(1)　当然，1℃のままです．

(2)　水とアルコールを混ぜると，一方の分子のすき間にもう一方の分子が入り込み，2lより少なくなります．

Quick BASIC

チョット出ましたサンカク野郎(三角関数)

オット，三角関数と聞いただけで目が三角に釣り上がってきた人はだれですか？　大丈夫です．あとで試験はありません．

ここでは，BASICの世界での三角関数について勉強して行きましょう．

三角関数は，ゲームの世界では重要です．計器の目盛板を書いたり，さまざまなところで必要になります．

では，BASIC流の三角関数の図を見てください．

SIN　サイン

A　B　X

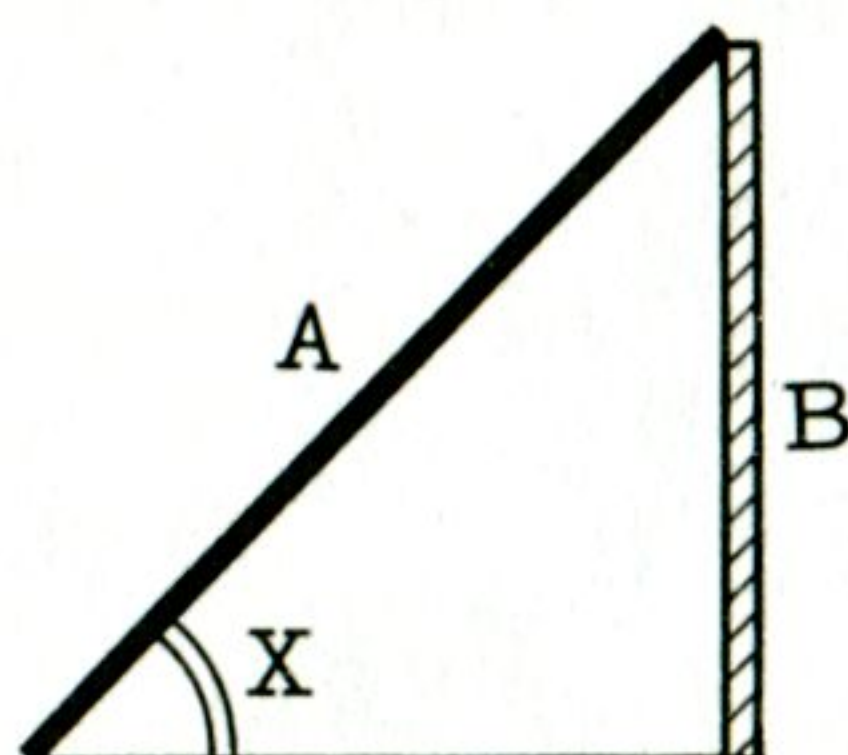
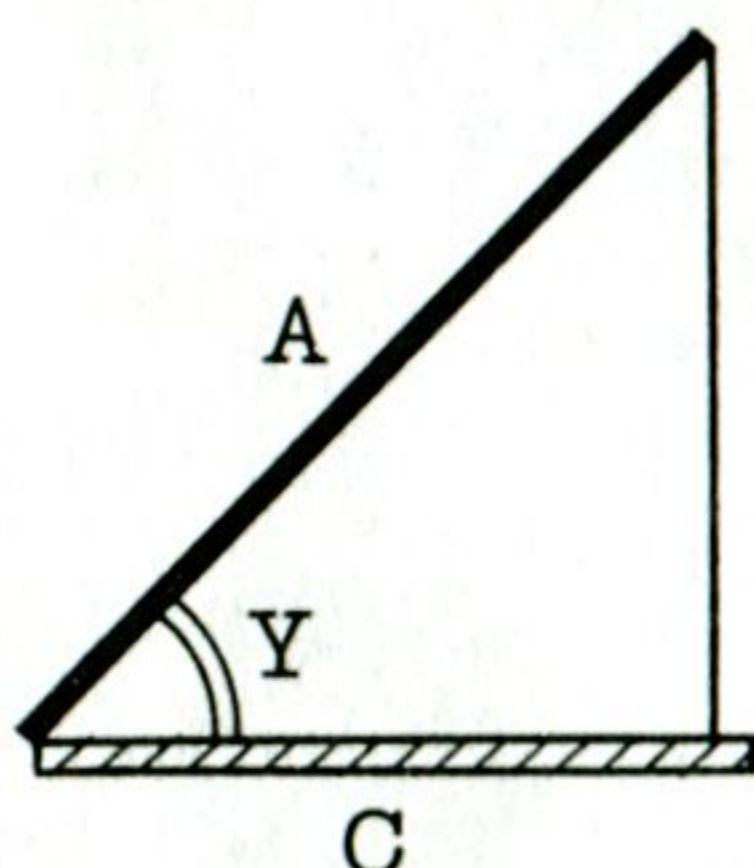
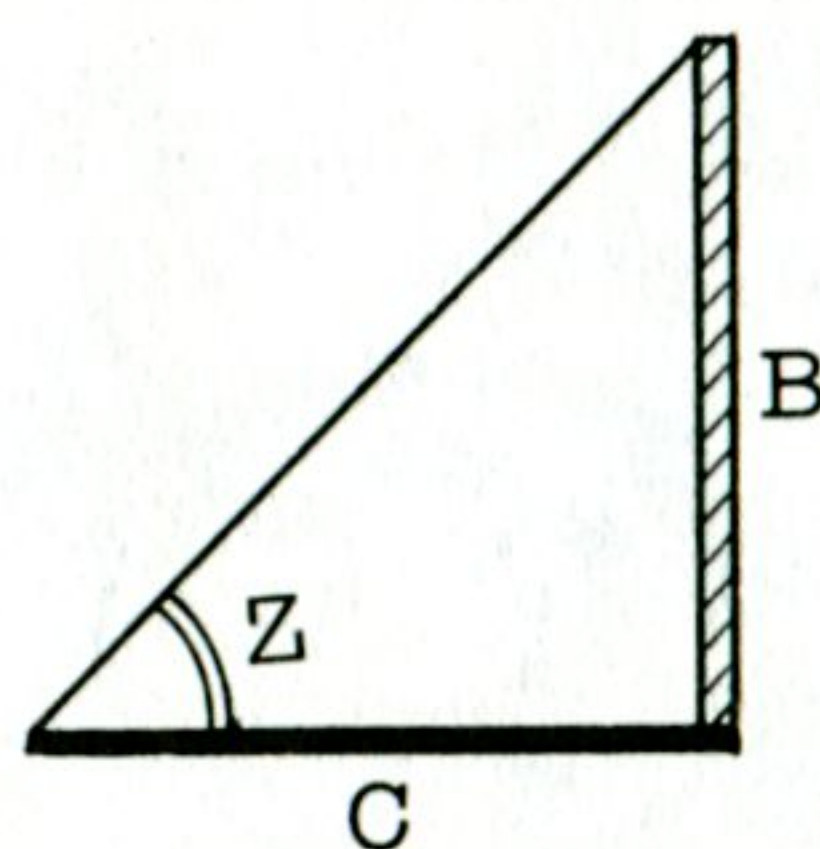
```
B_=_A_*_SIN(X)
A_=_B_/_SIN(X)
```

COS　コサイン

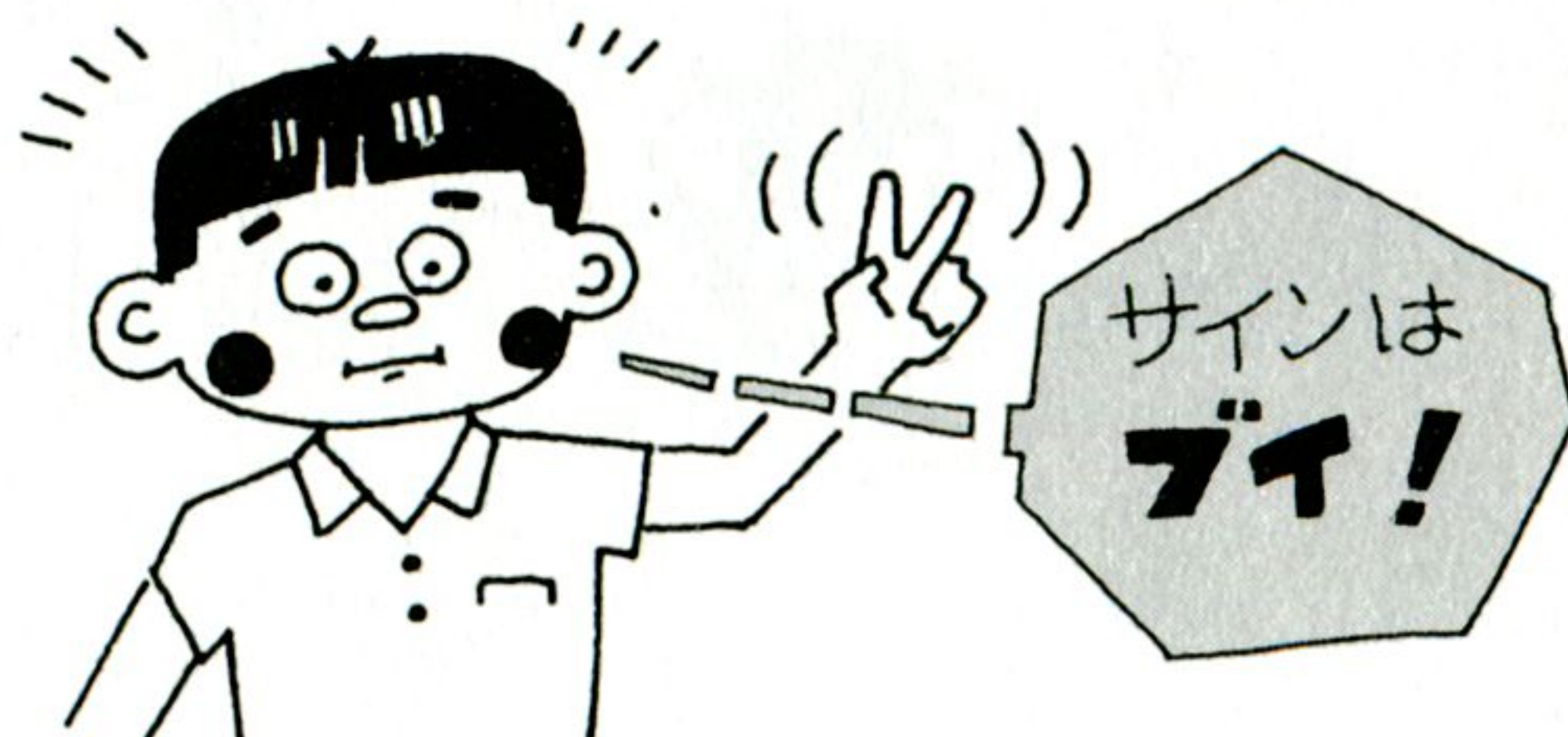
A　Y　C

```
C_=_A_*_COS(Y)
A_=_C_/_COS(Y)
```

TAN　タンジェント

B　Z　C

```
B_=_C_*_TAN(Z)
C_=_B_/_TAN(Z)
```

ただし，ここで気を付けなければいけないのは，『角度は例によってラジアンで表されている』ということです．つまり，360°を2π，つまり6.28…で示しています．

サインは
ブイ!

では，こで実際に三角関数を使って計器の目盛板を書いてみましょう．下のプログラムを入力，保存した後，実行してみてください．

METER．BASプログラム

```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
CIRCLE (200, 200), 100, 7
FOR J = 1 TO 50
    JX = 200 + 100 * SIN(J * 3.141592 * 2 / 50)
    JY = 200 + 100 * COS(J * 3.141592 * 2 / 50)
    LINE (200, 200)-(JX, JY), 7
NEXT J
CIRCLE (200, 200), 90, 5
PAINT (200, 200), 0, 5
CIRCLE (200, 200), 90, 0
END
```

METER2．BASプログラム

```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
CIRCLE (200, 200), 100, 7
FOR J = 1 TO 50
    JX = 200 + 100 * SIN(J * 3.141592 * 2 / 50)
    JY = 200 + 100 * COS(J * 3.141592 * 2 / 50)
    LINE (200, 200)-(JX, JY), 7
NEXT J
CIRCLE (200, 200), 90, 5
PAINT (200, 200), 0, 5
CIRCLE (200, 200), 90, 0
FOR J = 0 TO 50 STEP 5
    JX = 200 + 100 * SIN(J * 3.141592 * 2 / 50)
    JY = 200 + 100 * COS(J * 3.141592 * 2 / 50)
    LINE (200, 200)-(JX, JY), 7
NEXT J
CIRCLE (200, 200), 80, 5
PAINT (200, 200), 0, 5
CIRCLE (200, 200), 80, 0
END
```

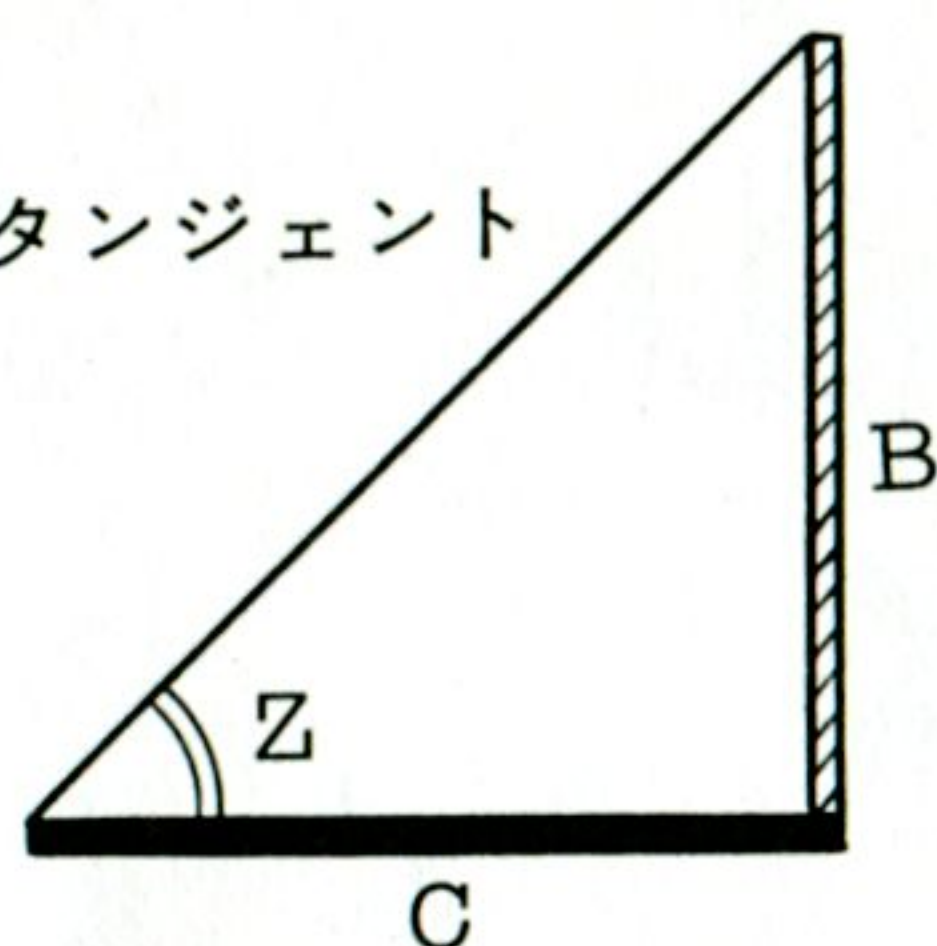
★もう１つの三角関数 ATN（アークタンジェント）

いままで出て来た三角関数は，すべて角度がわかっているときに，１つの辺の長さから他の辺の長さを求めるものでした．この ATN は，直角を挟む２つの辺の長さがわかっているときに，対応する角度を出すものです．

ATNアークタンジェント

B

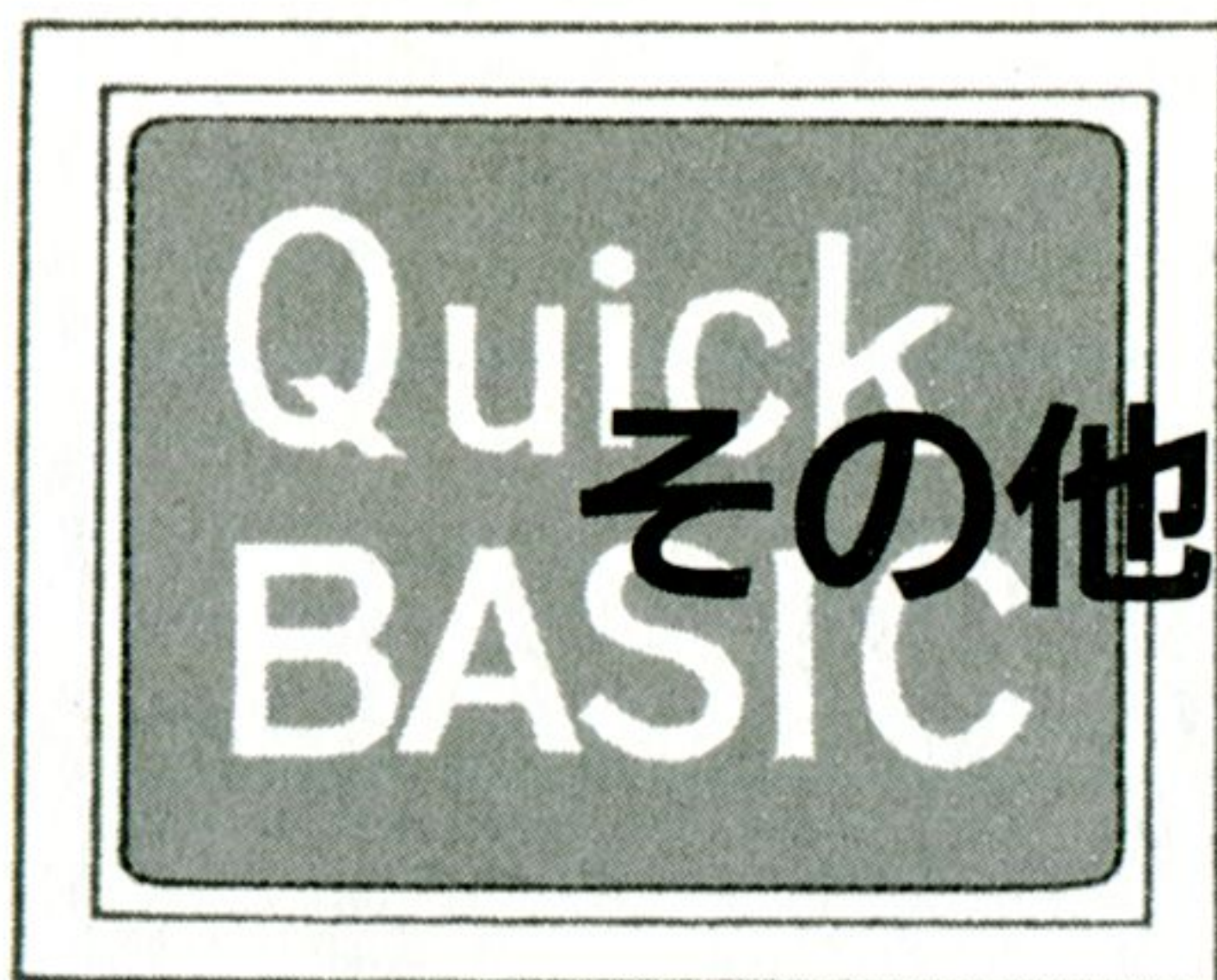
Z

C

Z_=_ATN(B_/_C)

もちろん，Zの単位はラジアンで出てきます．角度にしたい人は，180/3.1415 を掛けてください．

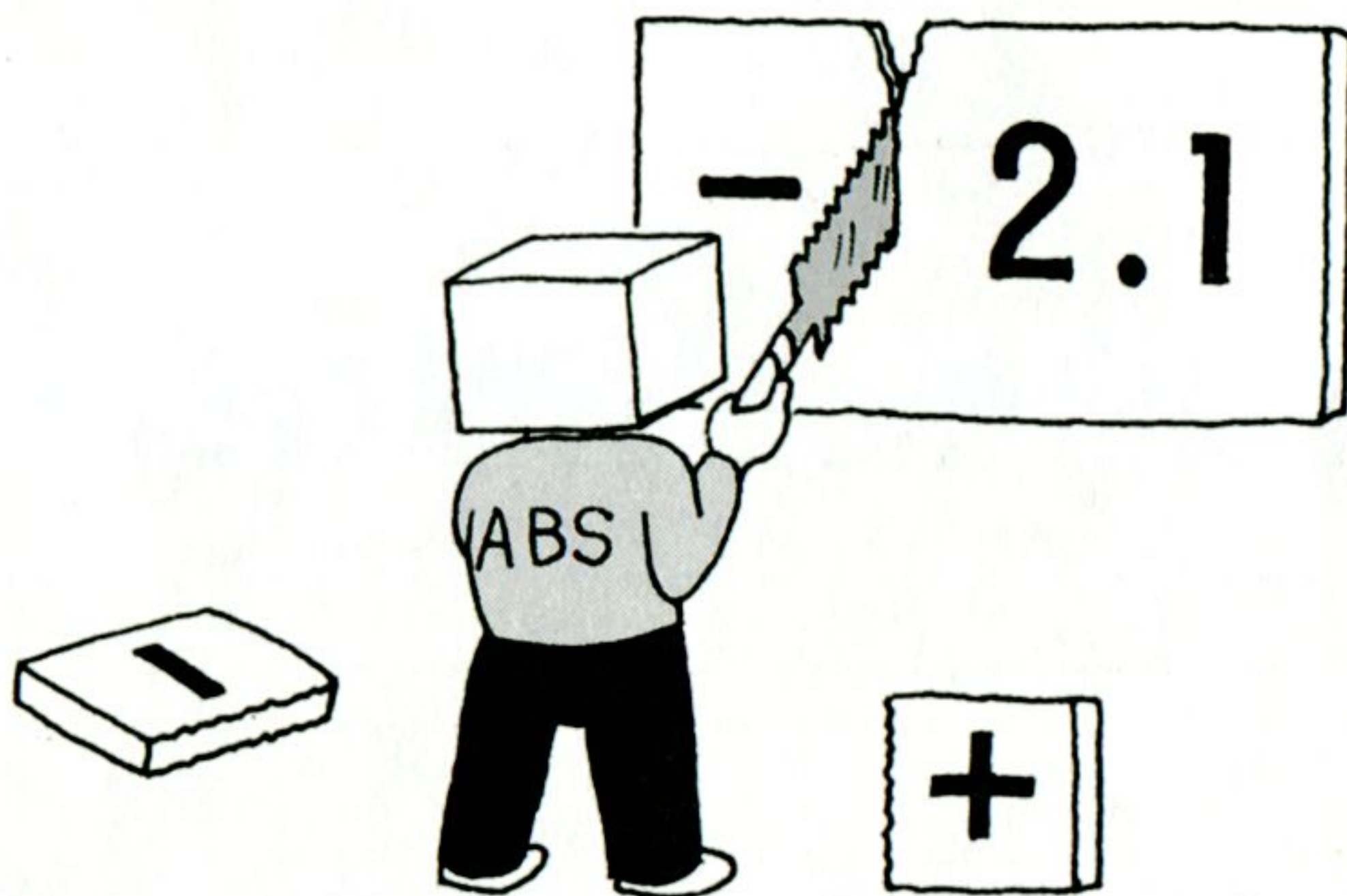
ATN

SIN COS TAN

チョイト出ましたサンカク野郎！

Quick BASIC その他の算術

Quick BASIC では，三角関数以外にも関数がいろいろと使えます．次に，それらを見ていきましょう．

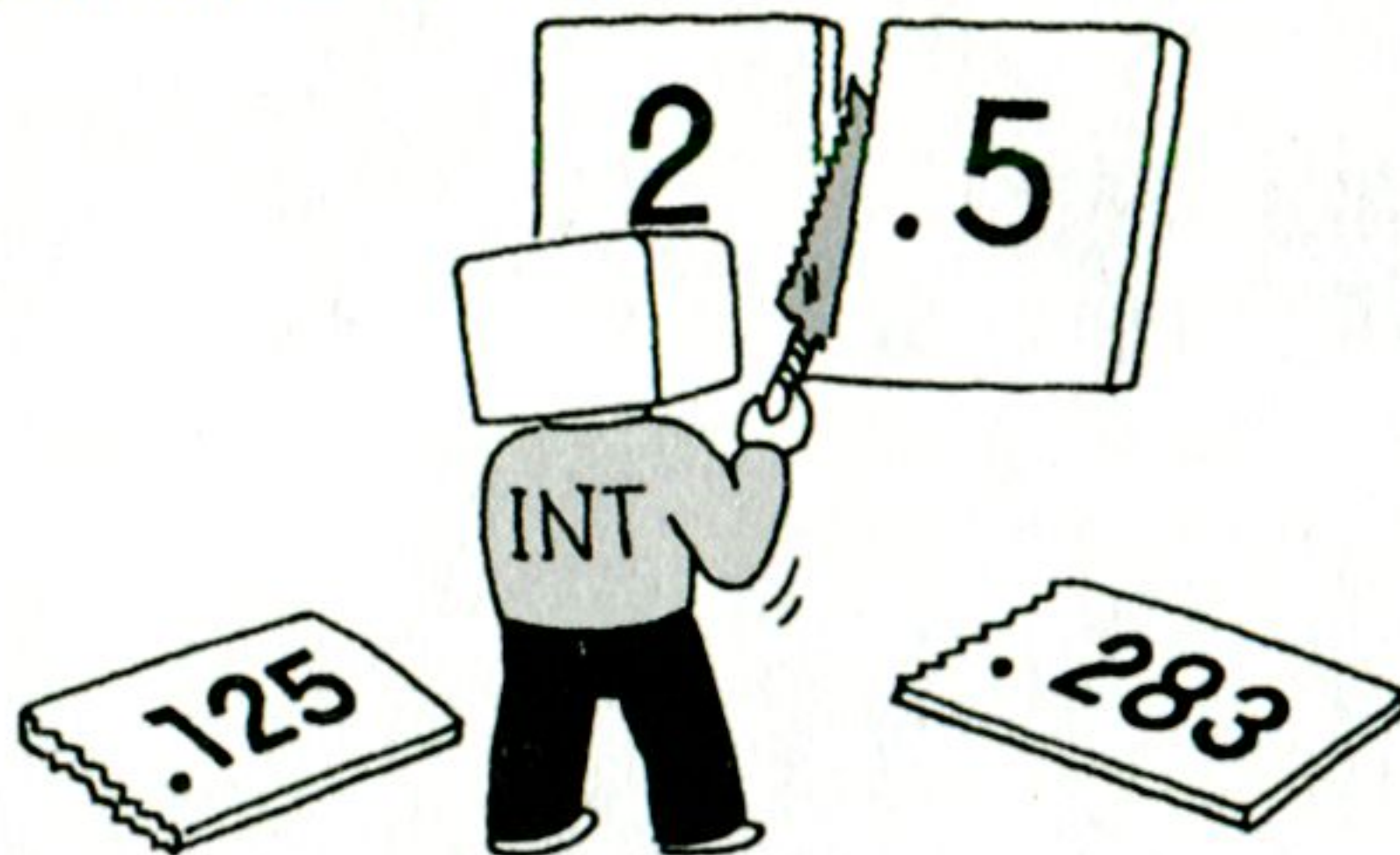
ABS（アブソリュート）

負の数・正の数ともに正の数に変換する関数です．次のように使います．

ABS

ＡＢＳ．ＢＡＳプログラム

```
CLS
A = 2.2
B = -2.2
C = ABS(A): PRINT C
D = ABS(B): PRINT D
END
```

イント
INT（インテジャー）

小数点以下を切り捨てて，その数より小さい整数を表す関数です．

```
INT．BASプログラム
  CLS
  A = 1.2
  B = -2.2
  C = INT(A): PRINT C
  D = INT(B): PRINT D
  END
```

フィックス
FIX

INTと似ていますが，負の数の場合，その数より大きい整数となってしまいます．単純に小数点以下の数字を消してしまうと考えると良いでしょう．

```
FIX．BASプログラム
  CLS
  A = 2.2
  B = -2.2
  C = FIX(A): PRINT C
  D = FIX(B): PRINT D
  END
```

ログ
LOG

eを底としたときのnの自然対数を求めます．

```
LOG．BASプログラム
  CLS
  A = 12
  Y = LOG(A)
  PRINT Y
  END
```

EXP（イクスポネンシャル）

自然対数の底 e(2.7182…)のべき乗を計算します.

ＥＸＰ．ＢＡＳプログラム
```
CLS
A = 12
Y = EXP(A)
PRINT Y
END
```

DATE$（デート）

内部時計の現在の日付けを示します.

ＤＡＴＥ．ＢＡＳプログラム
```
CLS
A$ = DATE$
PRINT A$
END
```

TIME$（タイム）

内部時計の現在の時間を示します.

ＴＩＭＥ．ＢＡＳプログラム
```
CLS
A$ = TIME$
PRINT A$
END
```

VAL（ヴァリュー）

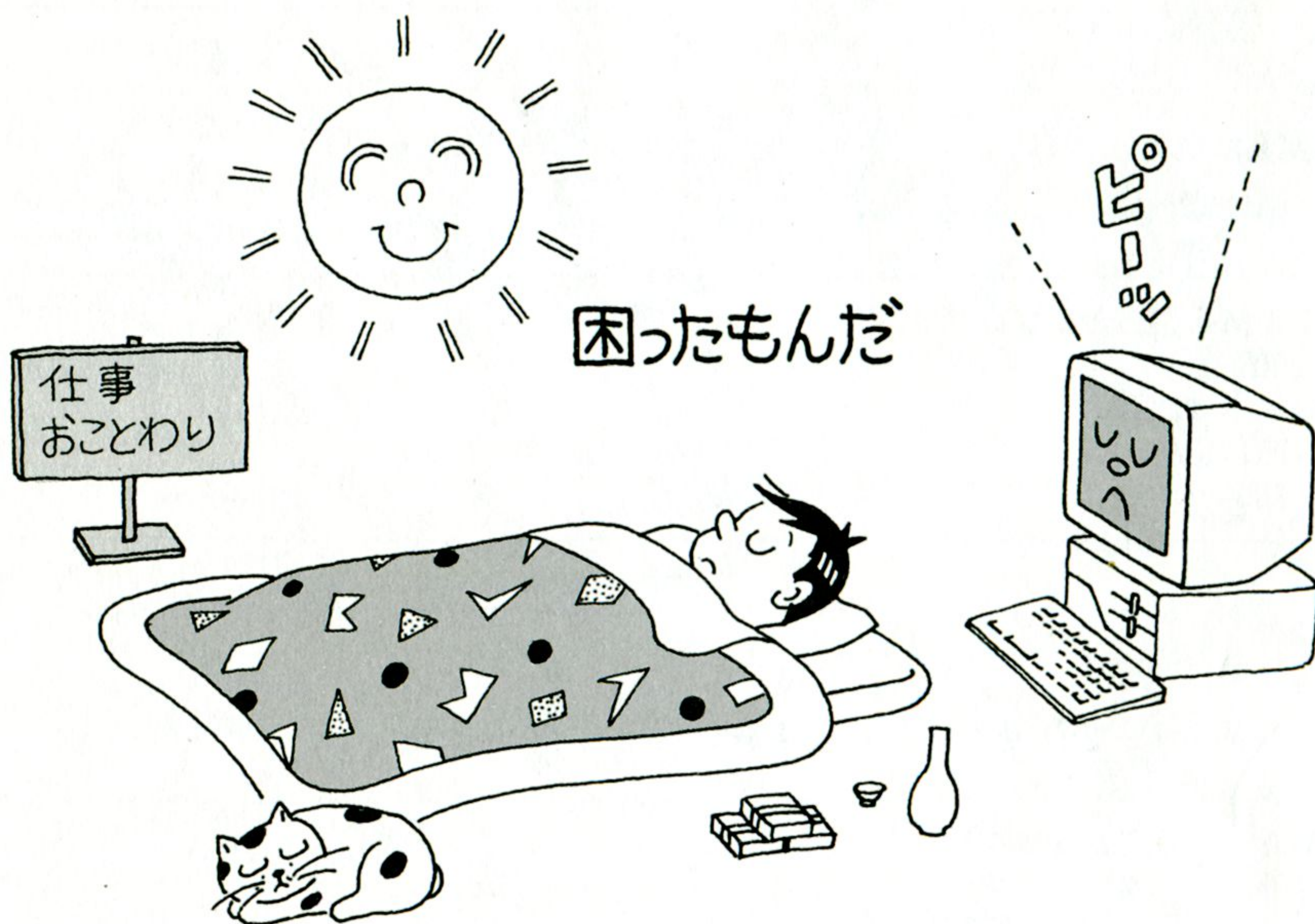
文字列として表現されているものを数字に直します.

ＶＡＬ．ＢＡＳプログラム
```
CLS
A$ = "123"
B = VAL(A$)
C = B + 456
PRINT C
END
```

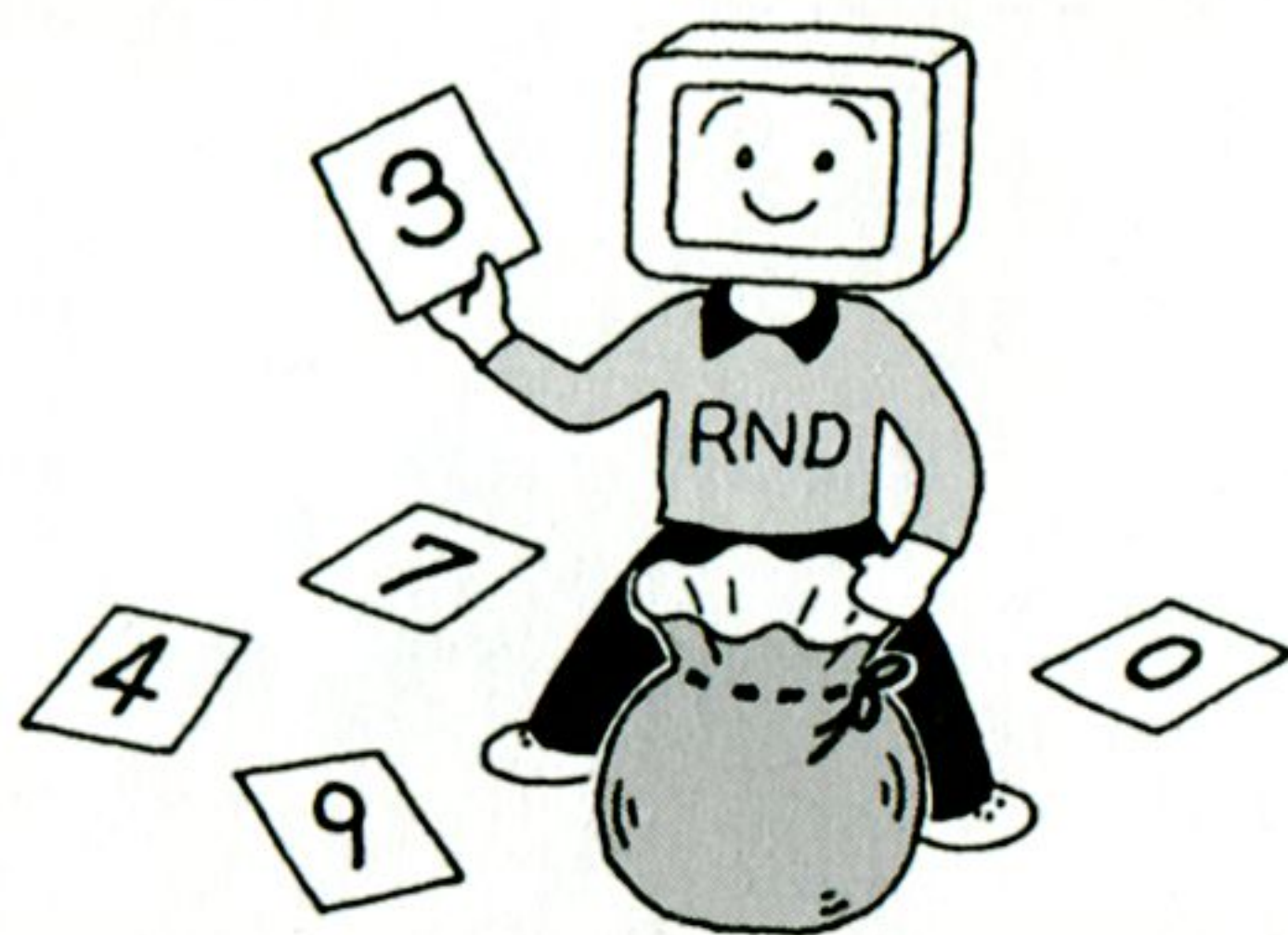
Quick BASIC でたらめって大好きです

目覚ましなんぞ気にせずに目が覚めるまで眠り続け，おなかがすいて来たらゴハンを食べる．気が向くまでは仕事はせず，そして，仕事も自分の好きなことしかしない，お金は好きなだけ勝手に使う．

ウーン，こういうでたらめな生活って大好きです．

エッ，それはでたらめじゃなくて怠惰なだけですって？　そういえば……．

RND

正確がモットーと思われるコンピュータの世界でも，アバウトでいいかげんなやつが意外とモテるんです．なんでそんなやつが？　と思っているんでしょうが，こんな使い方があります．まずシミュレーションがあげられます．人の流れなどさまざまな事象をコンピュータでやってみて，銀行のカウンタ数を決めたり，道路を設計したりするときなどがあります．こういう場合に，適当なでたらめが意外と有効なのです．また，これはゲームなどでは，とても大事なことなのです．いつも同じ場所で，同じ敵が出て，同じゴールドを持っているというのでは，ゲームがつまらなくなってしまいます．そういうときに，このでたらめが意外と役に立ちます．

Quick BASICの世界で，このでたらめを作るには，RND(ランダム)を使います．

RND

RND．BASプログラム

```
CLS
X = RND
PRINT X
END
```

このプログラムを，例によって実行してみてください．

RNDと書くと，書いた部分が0から1の間のでたらめな数となります．

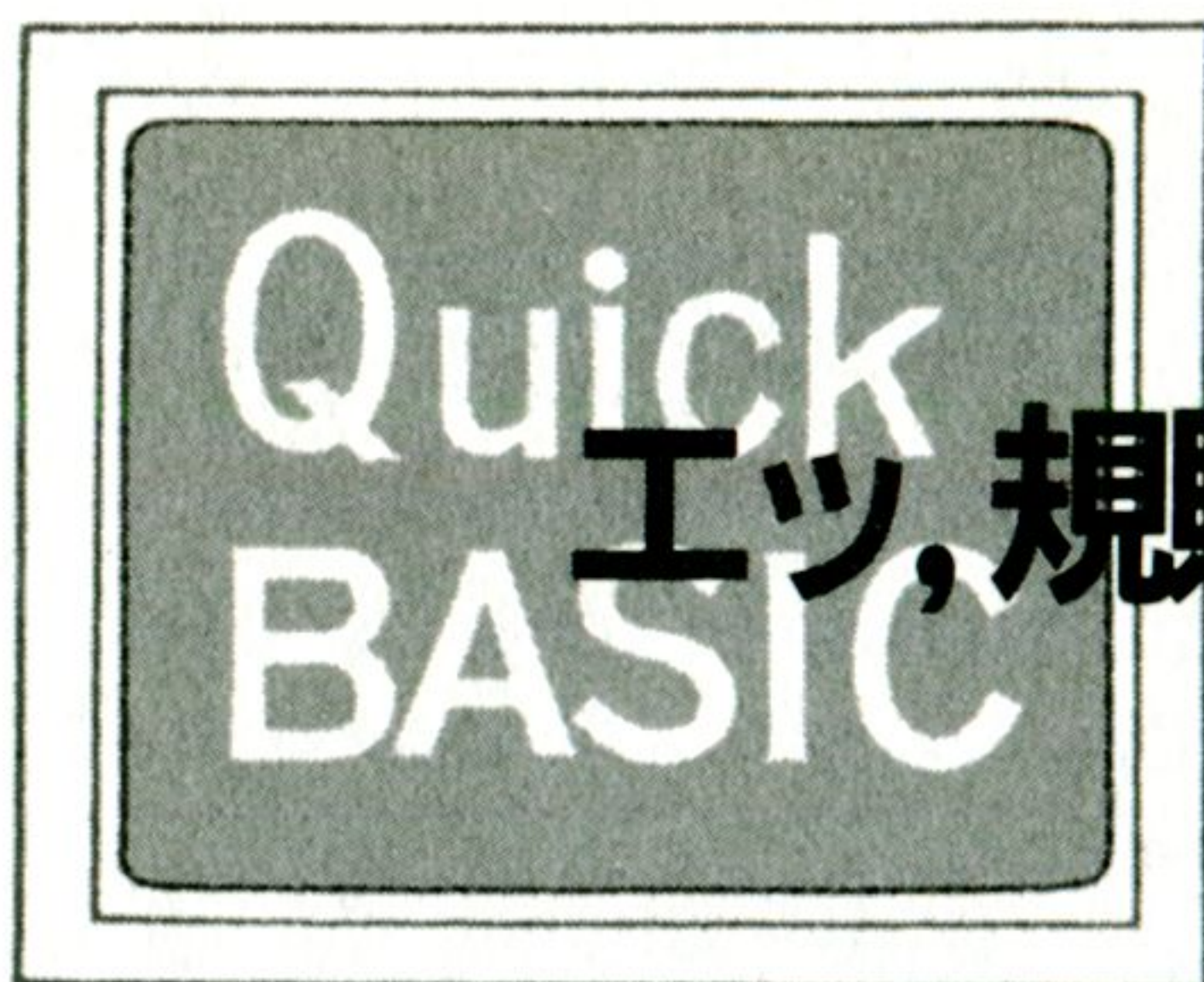
もし，0から1でなく0から365までの間のでたらめな数がほしいときは，

```
A_=_RND_*_365
```

のように替えてください．さらに，0から365までの間の整数がほしいときは，あとの(　　)の中の数を整数にする命令INT(インテジャ)を使って，

```
A_=_INT_(RND_*_365)
```

のように書いてください．

Quick BASIC

エッ，規則正しいでたらめって？

実は，前項のでたらめを得る関数 RND には，そのままではとんでもない欠陥があるのです．この RND だけでは，そのプログラムを実行するたびに，出て来る数が毎回同じなのです．つまり，プログラムの中では 0.9，0.1，0.5 とでたらめに出てくるのですが，もう 1 回同じプログラムを実行させると，また 0.9，0.1，0.5 と出てきてしまいます．これじゃあでたらめというのは“カンバンにいつわりあり”の“誇大広告”です．

実は RND は，ある計算式を使って一見でたらめそうな値を出しているだけなのです（コンピュータにとっては，でたらめなんてやっぱり不得意分野だったのですネェー）．

そこで，この式の元になる数を変えてやれば，また違った乱数を出してきます．この元になる数を変える命令が RANDOMIZE（ランドマイズ）です．

RANDOMIZE_5 のように，あとに −32768 から 32767 までの数を書くことで発生する乱数の元を変えられます．ただし，この行をそのままプログラムに書いたのでは，今度は毎回 5 を元にしたデタラメしか発生しません．そこで，本当のでたらめにするには次のプログラムを RND を使う前に入れてください．

```
RNDOMIZE．BASプログラム
  CLS
  SEED = INT((TIMER - 32767) / 1.5)
  RANDOMIZE SEED
  X = RND
  PRINT X
  END
```

このプログラムは，午前 0 時からの秒数を与える TIMER 変数を RANDOMIZE の元にしています．これなら，同時刻に始めない限りは，系列は違ってきます．

規則正しくないでたらめがほしい人は，プログラムの RND の出てくる前に，プログラム RANDOMIZE.BAS をそっくりまねして書いておいてください．

Quick BASIC

モジモジしないで(文字変数)

いまの女の子は，ウジウジしてるのが大きらいです．彼女の前でなにもいえずにモジモジしているのは，それだけでマイナスです．ダメでモトモト，ダメモト精神でアタックあるのみです．わが秘密探偵団によりますと，カワイイ子ちゃんほど，実はみんなが遠慮して，アタックしていないという結果が出ています．

ここで，モジモジならぬ，文字変数にアタックです．

Quick BASICでは，数字だけでなく文字も1つの変数として扱うことができます．そのとき大事なのは，変数のあとに$(ダラーマーク)を付けて，パソコンがこれは文字変数だとわかるようにすることです．

STRING．BASプログラム

```
CLS
A$ = "BOKU WA "
B$ = "ANATA GA SUKIDESU "
C$ = A$ + B$
LOCATE 3, 10
PRINT C$
END
```

いつものように，上のプログラムを入力し，保存してから実行してみてください．

A$_=_"BOKU_WA"

の行で，Aのあとの$が，A$は文字変数だということを示しています．次に，A$にしたい文字列は必ず"(ダブルクォーテーションマーク)で囲んでください．くれぐれも'(シングルクォーテーションマーク)と間違えないようにしてください．

文字変数は，このプログラムのように加えることもできます．もっとも＋（プラス）したからといって，中身が変わるわけではなく，2つの文字変数がつながるだけですが….

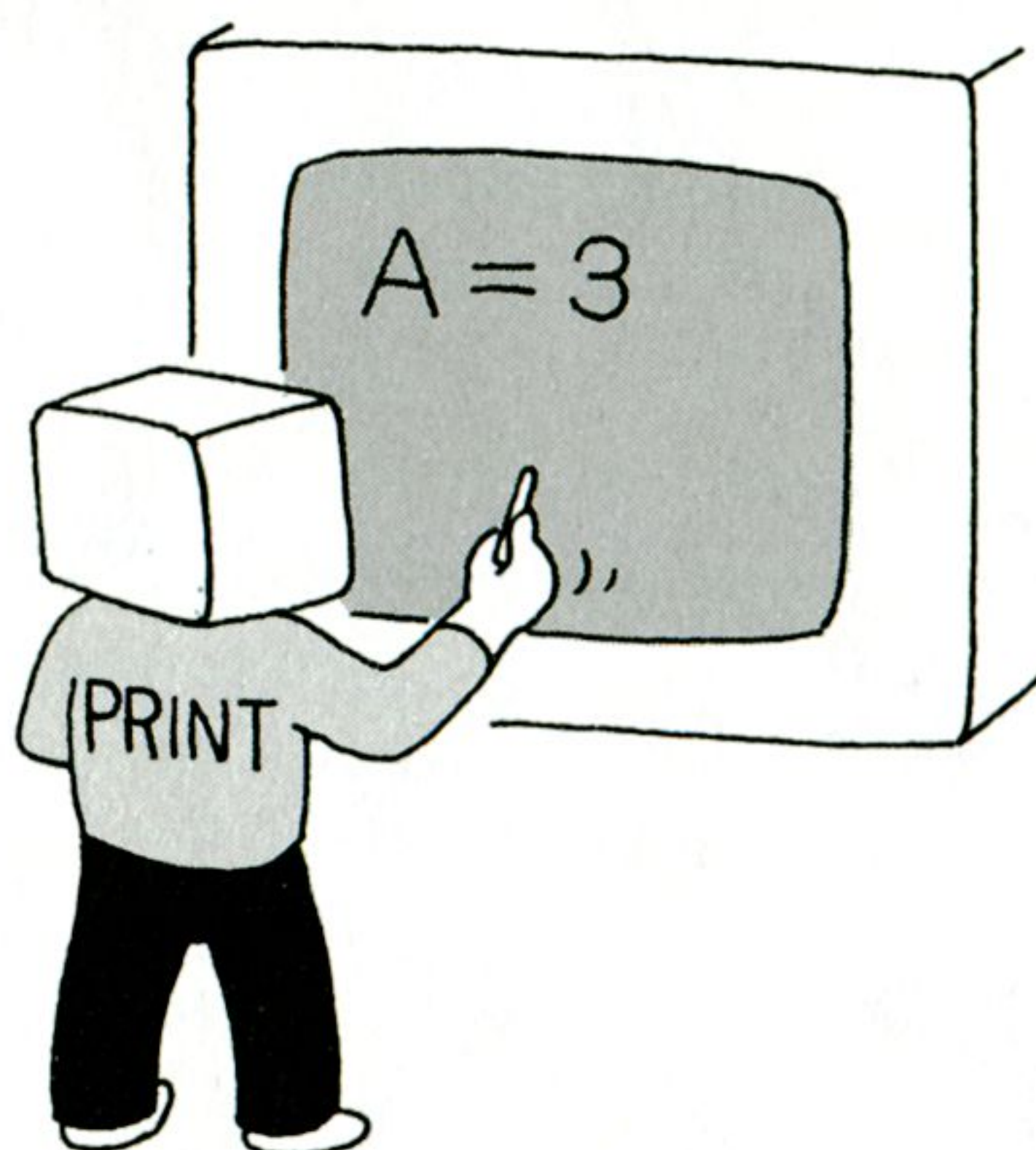
5行目の　LOCATE（ロケート）,_3,_10　の命令が，画面のどこに文字を書き始める起点を持って行くかの命令です．このプログラムでは，上から3行目，左から10文字目から始めるように指示しています．

そして，PRINT（プリント）文で画面に文字を表示させています．また，PRINT 文と”を使って，

ＰＲＩＮＴ．ＢＡＳプログラム

```
CLS
LOCATE 12, 15
PRINT "BOKU WA ANATA GA DAISUKIDESU "
END
```

このプログラムのように直接，望みの場所に文字を表示することもできます．

ここで，PRINT だと，いかにもプリンタに字を出せという命令のように見えます．実際は，PRINT は CRT の画面の上に字を出す命令です．プリンタに字を出させたい時には，LPRINT（エルプリント）　を使います．

コラム

LOCATE命令の違い

LOCATEは，PCシリーズのN88-BASICとQuick BASICでは大きく違う命令の1つです．

まず，Quick BASICでは，数字の前後のカッコがいりません．

次に，N88-BASICでの　LOCATE_(3,_10)　は，上から11行目の左から4字目からスタートすることを意味し，Quick BASICでは，まったく同じ命令が，上から3行目の左から10文字目というように行と列が入れ替わってしまい，かつ，行，列とも1つずつ減ってしまいます．これは，Quick BASICがIBM PCのBASICと互換性があるように作られているためです．

このため，N88-BASICで書かれたプログラムをQuick BASICに移植するときには，必ずLOCATEのあとの数字を入れ替えて，それから数字の値を調節してください．

これを行わないと，画面がグシャグシャになったり，プログラムを何度確認しても合っているのに，行数を超えてエラーが発生することがあります．

Quick BASIC たった７つで世界が開ける

★制御（コントロール）

野球のピッチャーにとってコントロールは命です．車の運転から原子力発電所まで，この制御が利かなければ，とんでもないことになってしまいます．

パソコンのプログラムにも制御構造があります．ときどきパソコンがまるで生きているかのように感じられるのも，この制御が良く考えてあるからでしょう．

プログラムの制御といっても，なにも難しいことはありません．『どこに行きなさい』とか，『こういう場合には……をしなさい』といった，流れを変える命令がいくつかあるだけです．

では，それらの命令を見ていきましょう．

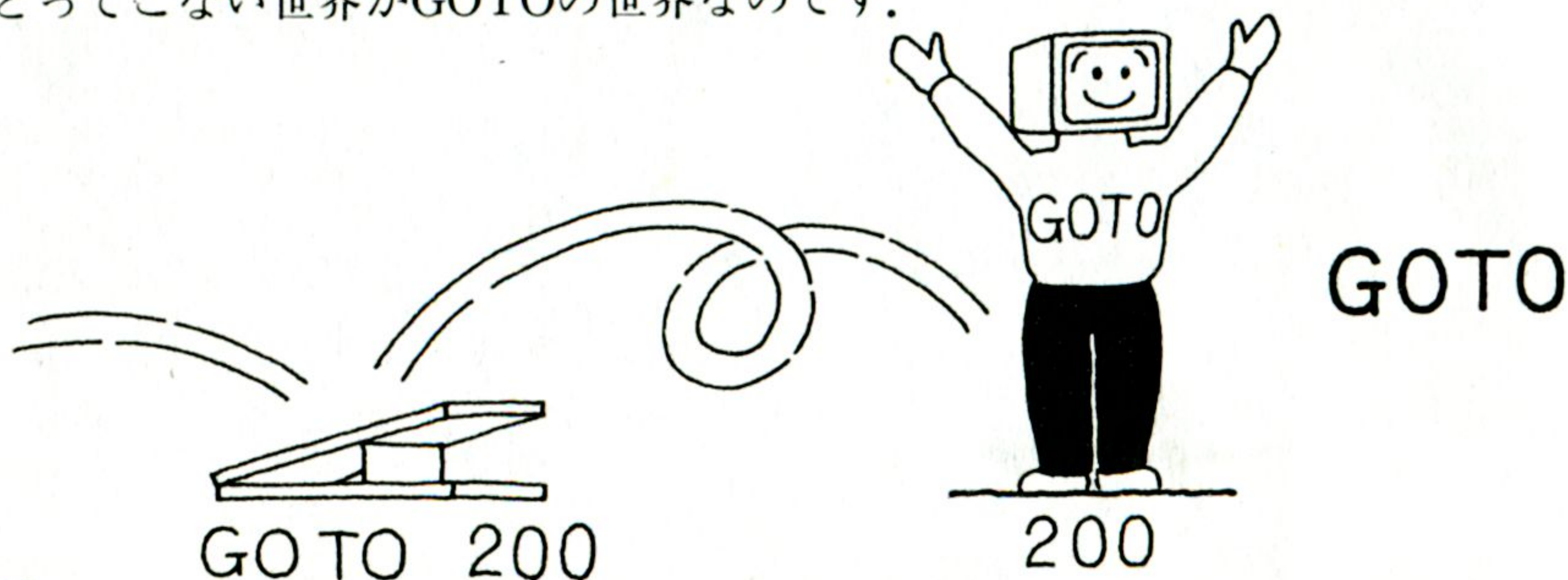
★“ふりだし”にもどる（スゴロクの世界）

子供のころ，お正月になるとスゴロクに熱中していました．やっとゴール近くまで上がってきたのに，あと１歩というところで“ふりだしにもどる”と書かれていたときのくやしさ，いまでも思い出します．その代わり，“95に行く”というようにビリだったのが，一挙大逆転したときの壮快感も忘れられません．

このように，あるところにくると，一度にどこかに飛んで行ってしまって，もどってこない世界がGOTOの世界なのです．

GOTO

GOTO 200

200

GOTO

ではまず，悪名高きGOTO命令です．この命令は，プログラムの世界では，まるでビリーザキッドかアルカポネか，すべての諸悪の根源のようにいわれています．これを使えば，プログラムがスパゲッティーのようになり，構造化と反対の極地にあるということになっています．そこまで悪いやつではないのですが，ともあれ，甘い誘惑であり，使いすぎると体に，オット，プログラムに悪いのも事実です．

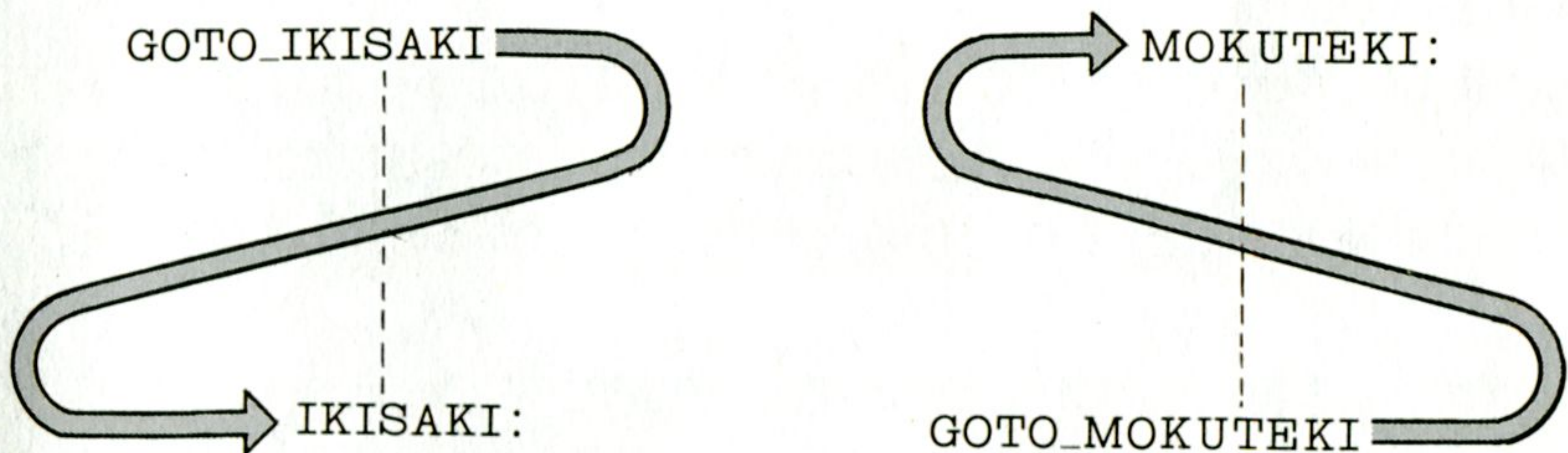
GOTOは，強制的に流れを指定先に変えてしまう命令です．通常は，あとに行番号を書き，GOTO_100　のように使います．

いままでのBASICでしたら，どの行にも必ず行番号が付いていましたので，その番号を書けばOKでした．Quick BASICでは，原則として行番号は入りません．プログラムは特別な分岐がない限り順に上から下に実行されていきます．そこでGOTO文を使いたい場合，自分の行きたい先の行の先頭になにか適当な番号を付けてください．この番号は同じ番号が2ヵ所にさえなければ，別に順番とかにこだわる必要はありません．

番号を使うと見通しが非常に悪くなるので，変わりに次のように描くこともできます．むしろこちらの方が望ましいともいえます．

GOTOってなに？

GOTO_IKISAKI
IKISAKI:

MOKUTEKI:
GOTO_MOKUTEKI

GOTOの自分の行かせたいところに，何か名前を付けます．目的地には，その名前とすぐその後に：(コロン)を付けたものをおきます．この行き先の文は，GOTO命令の前にあってもあとにあっても構いません．

GOTOのところまできたプログラムの流れは，強制的に目的のところまで飛んで行き，その行先からまたプログラムを実行していきます．

MOKUTEKI:

GO TO MOKUTEKI

エッ，帰りの燃料ないんですか？……

★1回休み

“すごろく”では，止まったところにより1回休み，芸をする，歌を歌うなど，他のことをやらなくてはいけないことがあります．他のことをやったあと，次の順番になれば，また同じように続けることができます．このように他のことをチョットやって終わったらもどってくるのがGOSUBの世界です．

GOSUB（ゴーサブ）

GOSUBは，GOTO文と似ている命令です．GOTOとの一番の違いは，GOTOではプログラムの流れがいったきりになりますが，GOSUBでは行き先で最後にRETURNと書くと，GOSUBの次の行にプログラムの流れがもどってくるところです．

プログラムを書いていると，いろいろなところで同じプログラムの部品（サブルーチン）を何回も使うことがよくあります．毎回毎回，同じプログラムを書くのはくたびれてしまいますし，メモリをくってしまいます．こういうときは，同じ仕事をする単位をサブルーチンとしてひとまとめにしておき，必要なときにいろいろなところから呼んだ方が便利です．

GOSUBってなに？

GOSUB_BUNGYOU
次の行

BUNGYOU:
ひとまとめにした仕事
RETURN

GOSB_BUNGYOU
次の行
BUNGYOU:
RETURN
GOSUB_BUNGYOU
次の行

同じルーチーンを違う場所から何回でも呼び出せる.

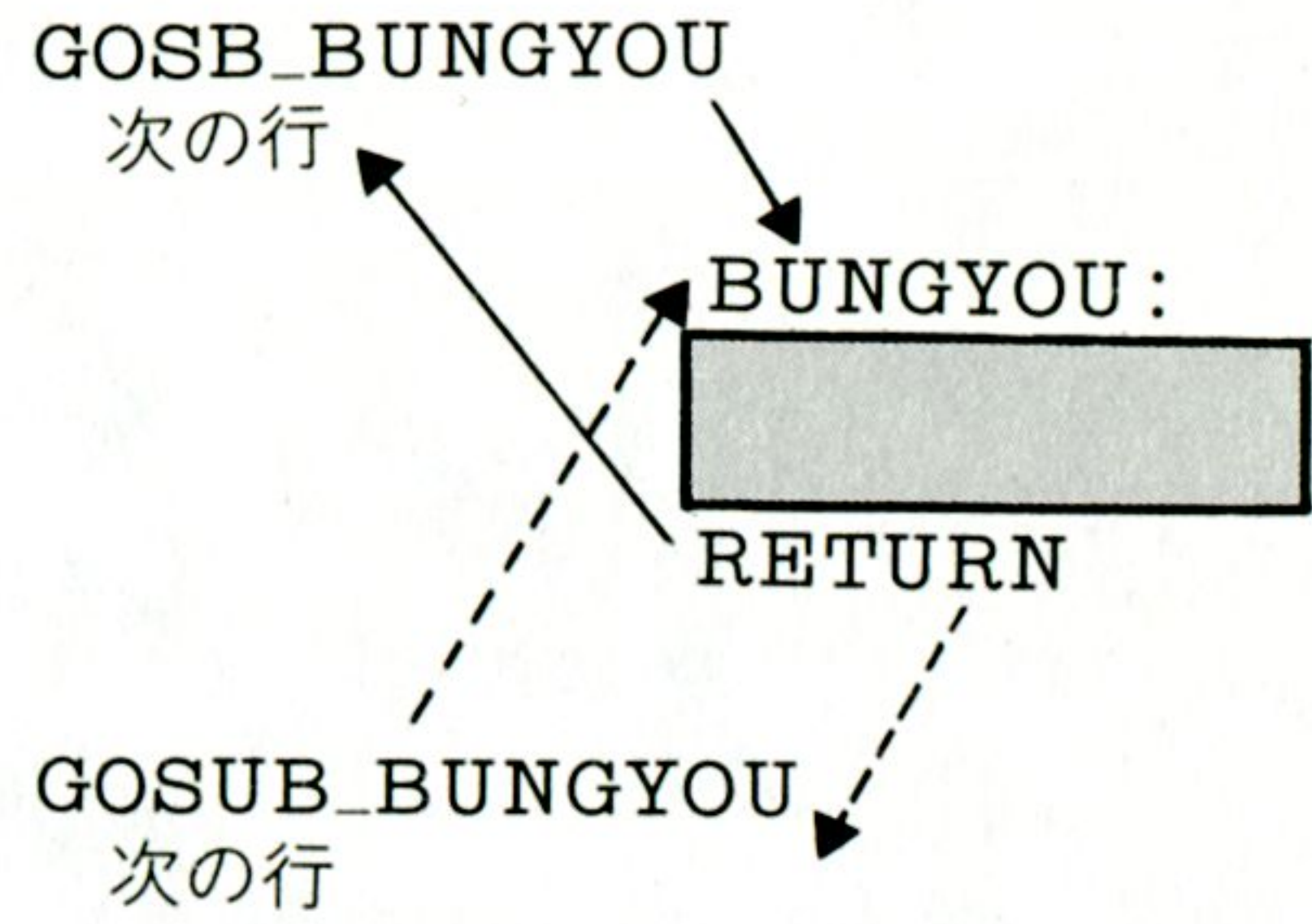
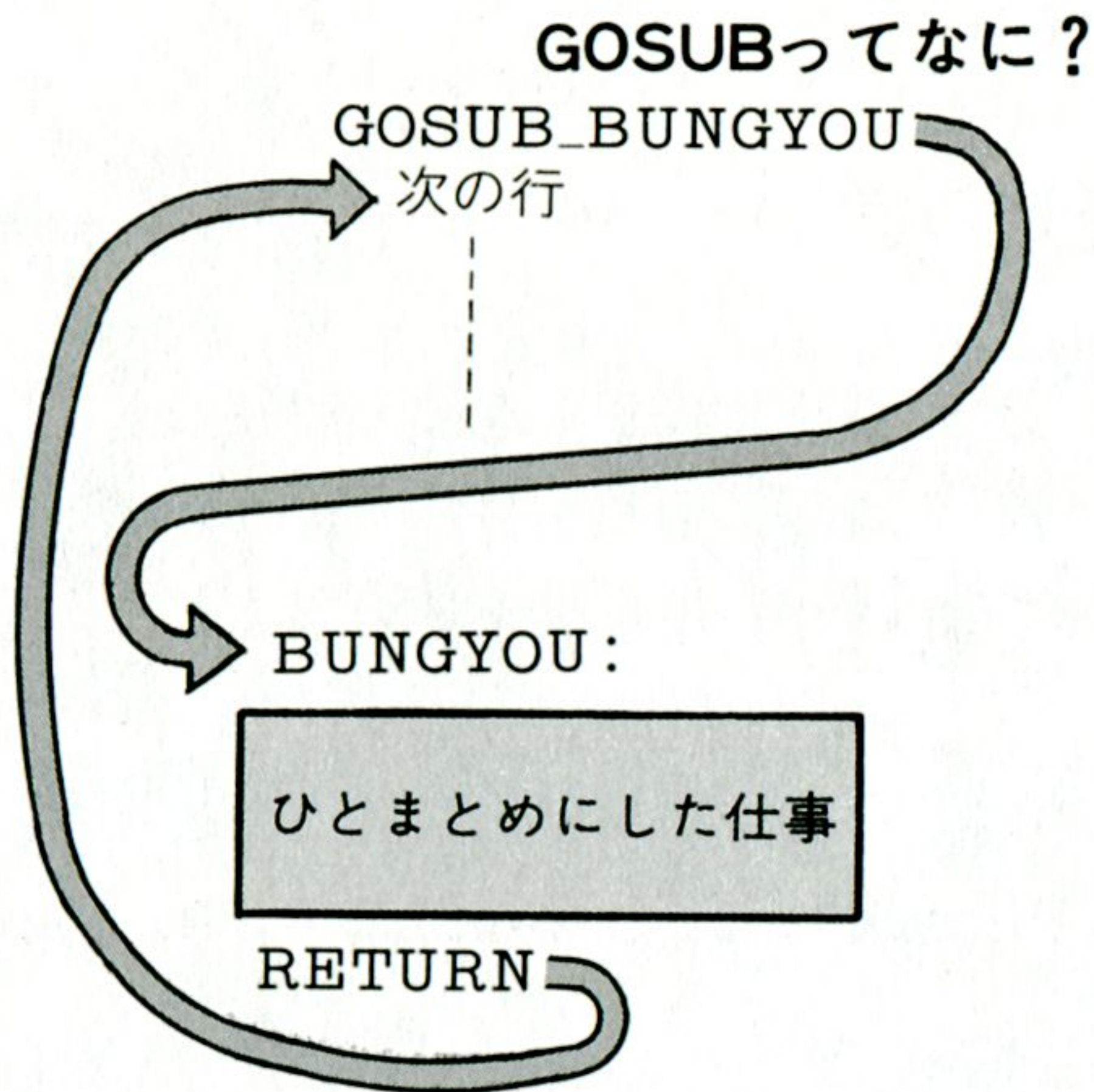
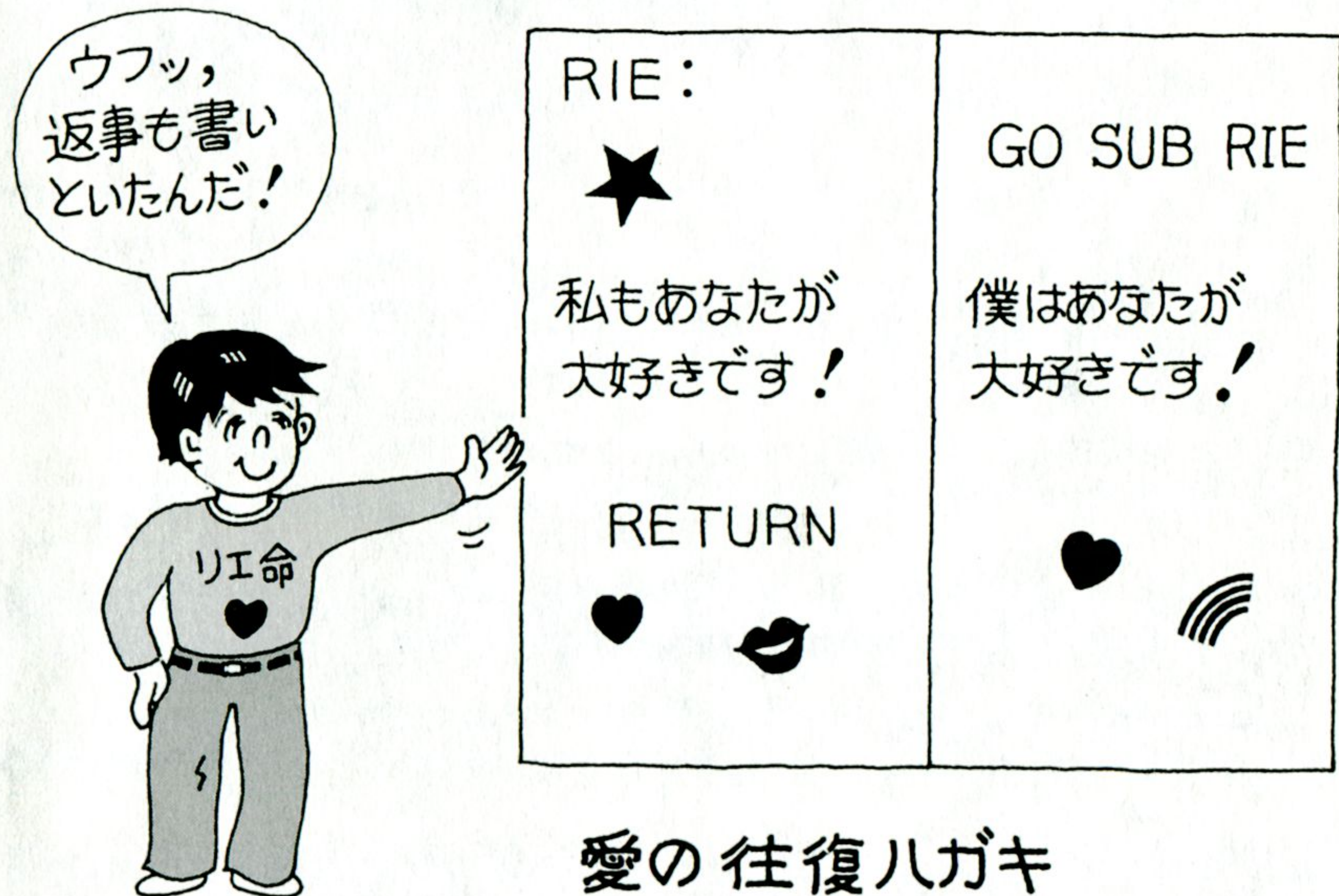
ウフッ，返事も書いといたんだ！

リエ命

RIE:
私もあなたが大好きです！
RETURN

GO SUB RIE
僕はあなたが大好きです！

愛の往復ハガキ

コラム

GOSUB の違い

Quick BASIC では，サブルーチンはSUBで宣言して本体と切り離した方が，メモリを有効に使えます．ただし N_{88}-BASIC やいままでのBASIC で書かれたプログラムを使う時は，GOSUB 命令をそのまま使った方が楽です．

このとき注意しなければならないのは，行番号で呼び出すときは，問題ないのですが，名前で呼び出すときは，次のような違いがあります．

GOSUBの違い

Quick BASIC

```
 |
GOSUB_KEISAN
 |
 |
END
KEISAN:
 |
 |
RETURN
```

N_{88}-BASIC

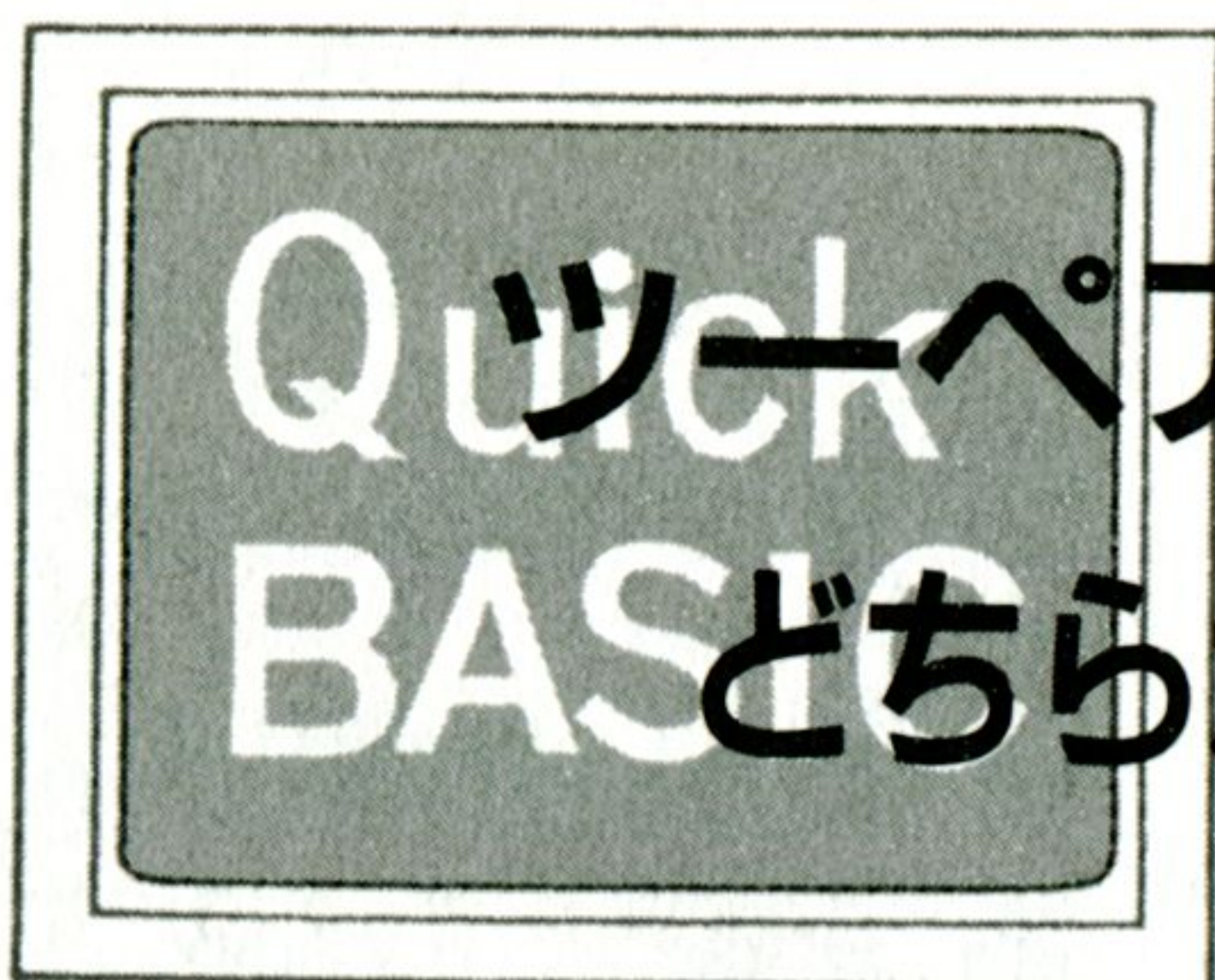
```
     |
100_GOSUB_*KEISAN
     |
     |
1000_END
2000_*KEISAN
     |
     |
3000_RETURN
```

N_{88}-BASICの方は，サブルーチンの名前の前に＊を付けて，＊KEISAN　のようにします．またこのサブルーチンを呼び出すときは，GOSUB_*KEISAN　のように呼び出します．

一方，Quick BASIC ではサブルーチンの名前の後に：(コロン)を付けます．つまり，KEISAN:のように直す必要があります．また，このルーチンを呼び出すときは，GOSUB_KEISAN　のようになにも付けません．

プログラムを移植するときには，気を付けてください．

プログラムのあとにサブルーチンをたくさん書き，サブルーチンの前にEND を忘れると「RETURN の所で対応するGOSUB がありません」というエラーメッセージが出て原因の発見に苦労します．END を忘れないようにしてください．

Quick BASIC

ツーペアとスリーカードはどちらが強い（ポーカーの世界）

ツーペアとスリーカードでは当然スリーカードの勝ちです．では，フルハウスとフォーカードでは？　ストレートフラッシュとフォーカードでは？　と考えていくと，わからなくなってしまいます．

しかし，キチンとルールが決まっていて，もし相手がフォーカードで自分がストレートフラッシュなら自分の勝ちというように決まっています．

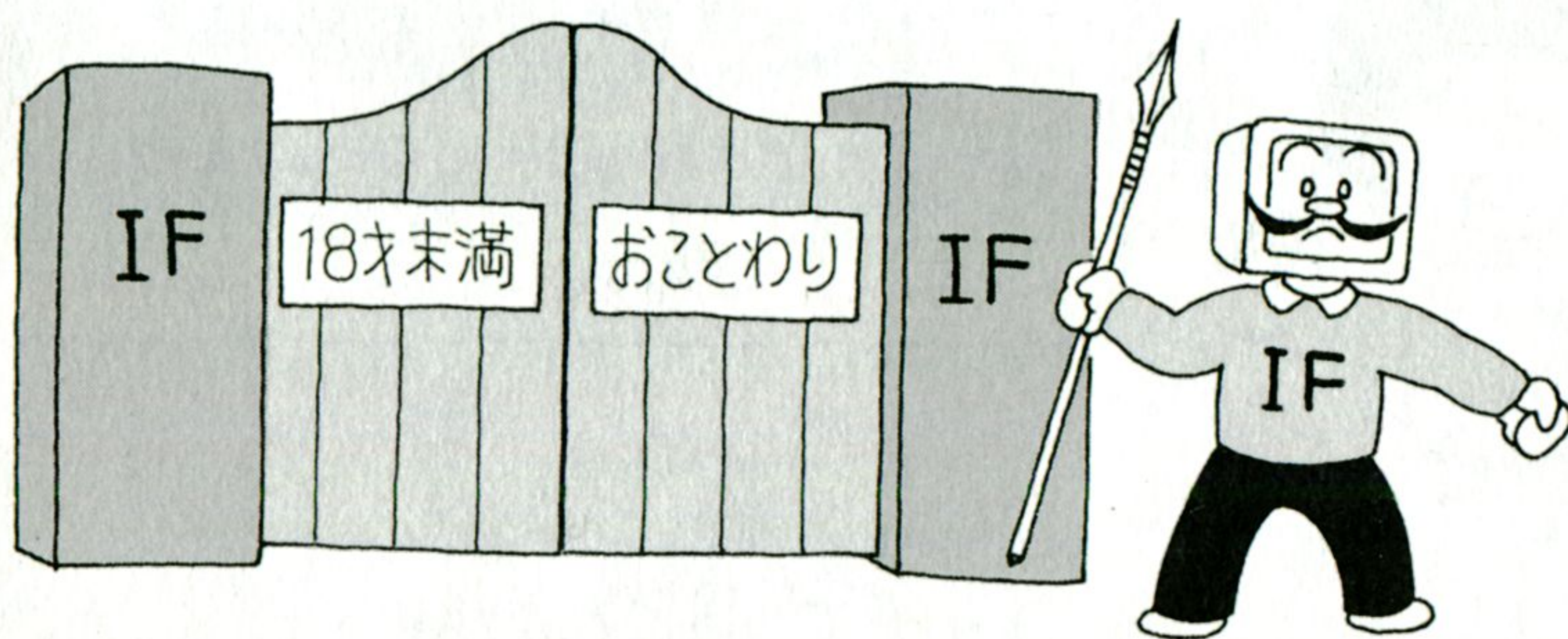
このように，『もし～ならば～しなさい』という命令がIF命令なのです．

もし，いま100万円あったら．もし『ユミチャン』がぼくの彼女だったら，もし……．

世の中には，たくさんの楽しい『もし』があります．または，『もしあの球が打てていれば』，『もしあのとき彼女に好きだといっていたら』というような苦い『もし』もあります．

パソコンの中での『もし』は，プログラムの流れを大きく変えることができます．

また，パソコンがあたかも知能があるようにふるまったり，さまざまな機械の制御ができるのも，この『もし』が上手に使えるからなのです．

IF

以下のプログラムを入力，保存したあとシフトキーを押しながらf･5キーを押して実行してください．

“INPUT_SUUJI”という字が現れたら，自分の入れたい数字を入れてリターンキーを押してください．

IF……THEN……

まず，IF文の一番の基本型式IF（イフ）……THEN（ゼン）…です．この文は，IF文のすぐあとの式が成立するならば，THENから後の文を実行しなさいという命令です．次のプログラムIF.BASを見てください．

```
IF．BASプログラム
  CLS
  PRINT "INPUT SUUJI"
  INPUT A
  IF A > 0 THEN PRINT "PLUS": PRINT "+": PRINT "0 YORI OOI"
  IF A < 0 THEN PRINT "MINUS": PRINT "-": PRINT "0 YORI CHIISAI"
  IF A = 0 THEN PRINT "ZERO": PRINT "0": PRINT "CHOUDO 0"
  END
```

これは，キーボードから数字を入力して，その後，リターンキーを押すと入力した数字が0より大きいかどうかを表示するプログラムです．

INPUT_AでAという数字を入力し，次の行からの IF_A_>_0 で，もし入力した値が0より大きいならばTHEN以下の PRINT_"PLUS" 以後を実行するようにしています．この例のように：（コロン）でつないで実行させたいことを次々に書いて行くことができます．

THENのあとで実行させたい文が複雑だったり，長かったりすることがあります．このような場合，先程のように1行で書くのは大変です．そこで，Quick BASICでは，次のような書き方もできるようになっています．次のプログラムを見てください．

ＩＦ２．ＢＡＳプログラム

```
CLS
PRINT "INPUT SUUJI"
INPUT A
IF A > 0 THEN
    PRINT "PLUS"
    PRINT "+"
    PRINT "0 YORI OOKII"
END IF
IF A < 0 THEN
    PRINT "MINUS"
    PRINT "-"
    PRINT "0 YORI CHIISAI"
END IF
IF A = 0 THEN
    PRINT "ZERO"
    PRINT "0"
    PRINT "CHOUDO 0"
END IF
END
```

この場合，IF_A_>_0_THEN　までで行を変え，その後の行にAが0より大きいときにやらせたいことを次々に複数行にわたって書いて行きます．このままでは，どこがこのIF文の終わりかがわからないので，最後にEND IFを付けて，ここでA>0に関する部分が終わりだよということをパソコンに教えてあげます．あとは，条件の分だけIF文を作っていけば良いのです．

第一の型と第二の型を混在させた次のような書き方は認められません．

```
IF A > 0 THEN PRINT "PLUS"
PRINT "+"
END IF
```

いくつもの条件で分けるときに，IF文を並べるのではなく，次のような書き方をすることができます．

```
IF3．BASプログラム
 CLS
 PRINT "INPUT SUUJI"
 INPUT A
 IF A > 0 THEN
     PRINT "PLUS"
     PRINT "+"
     PRINT "0 YORI OOKII"
 ELSEIF A < 0 THEN
     PRINT "MINUS"
     PRINT "-"
     PRINT "0 YORI CHIISAI"
 ELSE
     PRINT "ZERO"
     PRINT "0"
     PRINT "CHOUDO 0"
 END IF
 END
```

ELSEIF（エルスイフ）で，次の条件をチェックします．このELSEIFの文は，条件に応じていくつあっても構いません．最後にELSE（エルス）以後の文で，それまでの条件にすべて合わなかったときの処理をさせています．

コラム

IF文の注意点

パソコンの中での計算は，使えるビット数に限りがあるため，複雑な計算のあとで，ほんのわずかずれることがあります．このようなときに，たとえば，IF_A_=_20.0_THEN_END　のようなプログラムを書くと，Aが19.001　の次に　20.001のような値を取ると，正しく分岐できなくなってしまいます．

そこで，Aが順に増えて行き，ほぼ20になったときに流れを変えたい場合，IF_A_>_=_19.8　のように書いておいた方が安全です．

SELECT CASE

1から10までの処理のうち，どれか1つを選択する場合とか，出て来る結果が決まったいくつかの整数の値の1つになる場合などは，IF文を使うよりも，SELECT CASEを使った方がすっきりする場合があります．

次のプログラムを入力，保存して実行してみてください．

```
CASE.BASプログラム
 CLS
 INPUT "10 ﾏﾃﾞﾉ ｽｳｼﾞｦ ｲﾚﾃ ｸﾀﾞｻｲ  ", N
 SELECT CASE N
     CASE 1, 3, 5, 7, 9
         PRINT "KISUU"
     CASE 0, 2, 4, 6, 8, 10
         PRINT "GUUSUU"
     CASE ELSE
         PRINT "SONOTA"
 END SELECT
 END
```

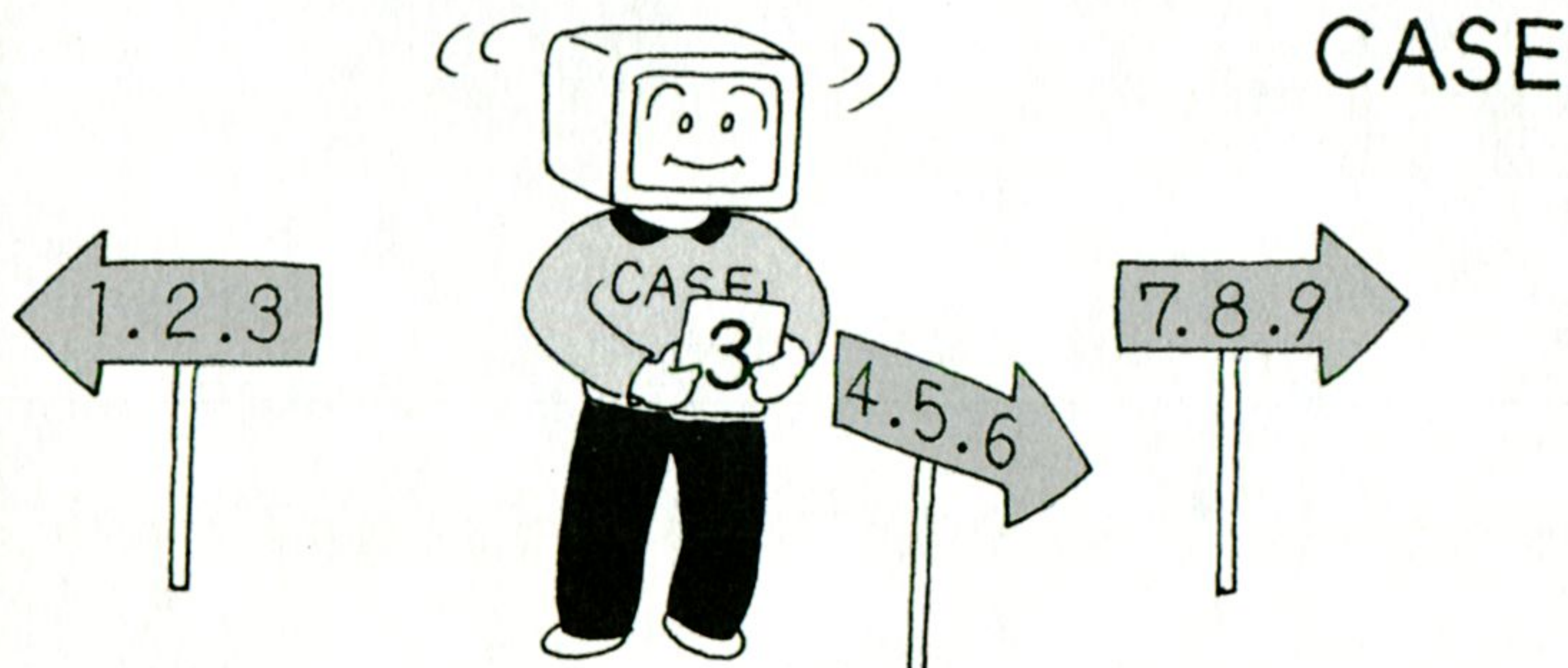
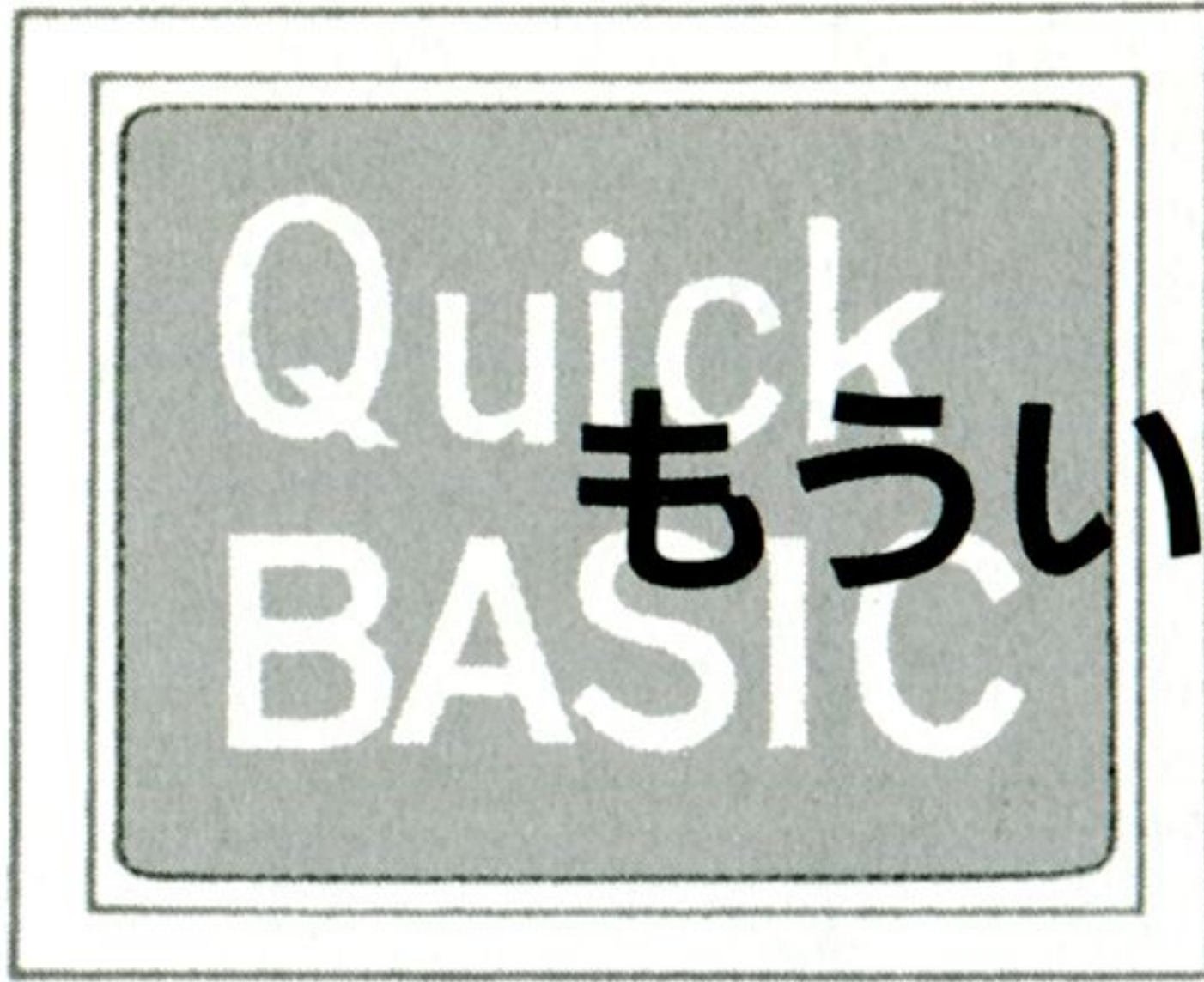
実行したあと，10までの数字を入れてリターンキーを押すと，奇数か偶数かを表示します．

プログラム4行目および6行目のCASEのあとに書かれている数字と，Nの値が一致するときにCASE文のあとに書かれた処理を実行します．CASEのあとに書く数字は，その処理に該当する場合をいくつでも並べて書くことができます．また，順番が違っても構いません．

プログラム8行目のCASE_ELSEからの行で，Nがそれまでの CASEのどれにも該当しなかった場合の処理を決めています．

INPUT文は，CASE.BASのようにすぐあとに"で囲んだ文を書き，その文を表示しながら入力を求めることができます．

カタカナを入力するには，カナキーを押してから字を入れます．

Quick BASIC

もういいかい，まあだだよ

子供のころ“カクレンボ”をしていて，うまく隠れ過ぎてだれも見付けてくれず，夕方になってしまった思い出があります．いまでは，とっても懐かしい思い出です．このカクレンボの世界が，DO（ドゥ） UNTIL（アンティル） と DO（ドゥ） WHILE（ワイル） の世界なのです．

DO UNTIL

“カクレンボ”をすると，最初のころは『もういいよ』というまで鬼はじっと待っています．この“～するまでは～していなさい”という世界が DO UNTIL の世界です．

DO UNTIL，だれかが『もういいよ』というまで待つ
『もういいかい』と聞く
LOOP（ループ）

この例でわかるように，UNTIL のあとに書かれたことが起こるまで，LOOP と書かれたところまでの間を繰り返し行います．

```
DOUNTIL．BASプログラム
  CLS
  INPUT "100 ﾏﾃﾞﾉ ｽｳｼﾞｦ ｲﾚﾃ ｸﾀﾞｻｲ  ", N
  GOUKEI = 0
  KAZU = 0
  DO UNTIL N < KAZU
      GOUKEI = GOUKEI + KAZU
      KAZU = KAZU + 1
  LOOP
  PRINT GOUKEI
  END
```

このプログラムは，0 からある数までの整数の合計を求めるプログラムです．

最初はKAZUを0としておいて，LOOPを1回通過するごとにKAZUを1つ増やし，その数をいままでの合計に加えて行きます．ここで，　N_<_KAZU　最初に入力した数NよりKAZUが大きくなるまでDOとLOOPの間を繰り返し行います．

パソコンの内部では，ある条件式が成立するときは1，成立しないときは0として扱っています．

DO UNTILの場合は，UNTILのあとに書かれた式が成立するまで，つまり，式全体の評価が1となるまでLOOPとの間を繰り返し実行します．

ここで，試しに　DO_UNTIL　のあとを数字の1だけを入れて　DO_UNTIL_1　のように替えてみてください．UNTILのあとの評価が1のため，LOOPまでの間は1回も行われず，プログラムが終わってしまいます．

では，　DO_UNTIL_0　とするとどうなるでしょうか？　パソコンは，あとの値が1になるまでやり続けるために，どこかで扱える数の範囲を超えるかSTOPキーを押すまではLOOPまでの間をやり続けることになります．

DO WHILE

“カクレンボ”もなれてくると，『もういいよ』という声の方向で，大体の場所がわかるようになってしまいます．こうなると，おもしろくなくなってくるので，子供心にも方法を考え，次の段階のやり方を工夫します．つまり，『まあだだよ』という声がしている間は待っているけれども，その声がしなくなったら探し始めるというやり方です．

“～している間は～する”という世界がDO WHILEの世界です．

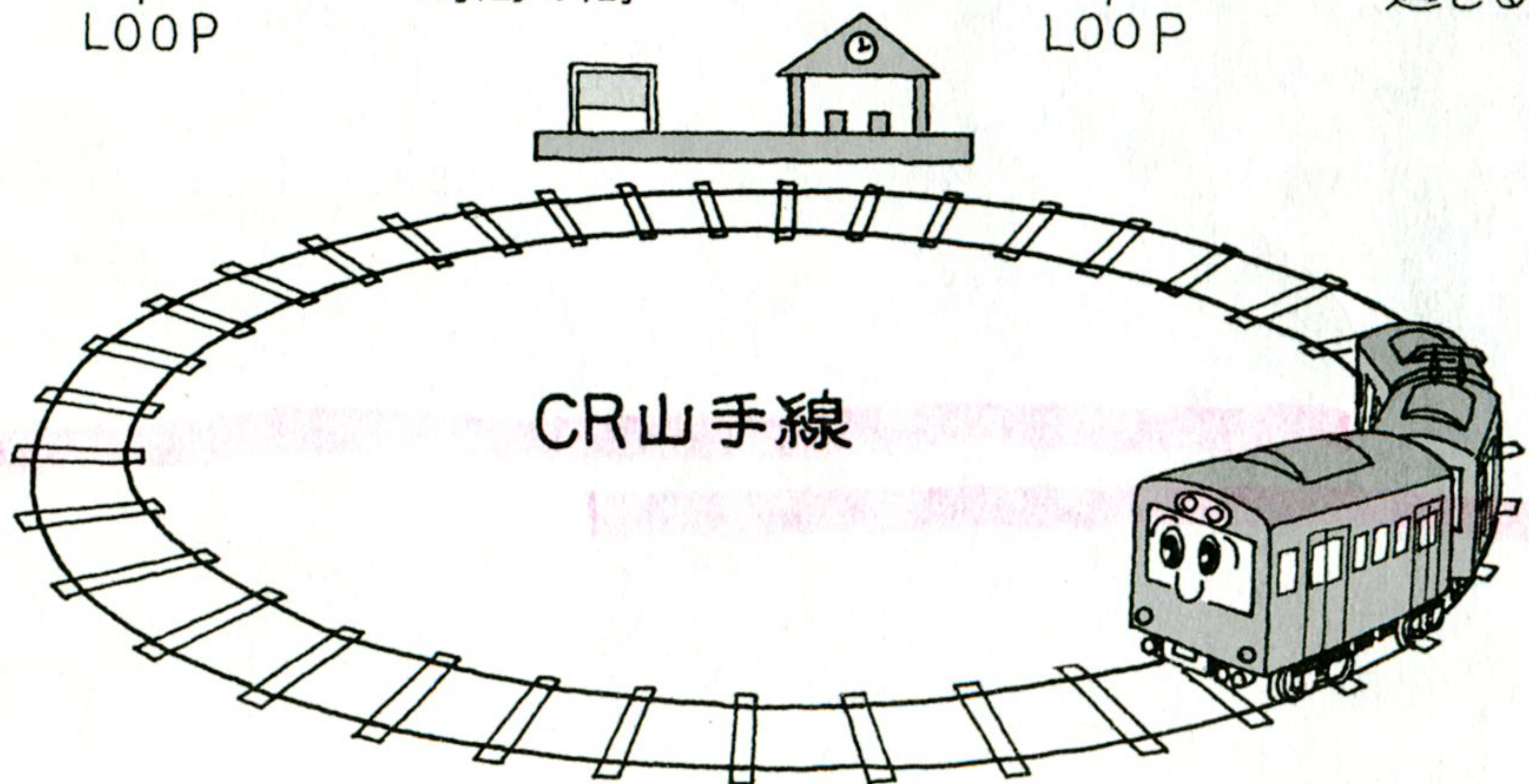
DO WHILE『まあだだよ』という声がしている間は待つ
　『もういいかい』という
LOOP

つまり，WHILEのあとに書かれたことが起こっている間は，LOOPとの間に，はさまれたことを繰り返し行っていきます．

```
DOWHILE．BASプログラム
  CLS
  INPUT "100 ﾏﾃﾞﾉ ｽｳｼﾞｦ ｲﾚﾃ ｸﾀﾞｻｲ  ", N
  GOUKEI = 0
  KAZU = 0
  DO WHILE KAZU <= N
      GOUKEI = GOUKEI + KAZU
      KAZU = KAZU + 1
  LOOP
  PRINT GOUKEI
  END
```

DO WHILEのプログラムは，先程のDO UNTILのプログラムとほとんど同じです．1つだけ違うのは，DO_WHILE_KAZU_＜_＝_N となっている部分です．

先程のプログラムと違い，DO WHILEはあとの式の評価が1のときにLOOPとの間を実行し続け，評価が0になるとき，つまり，あとに書いてある式が成立しなくなったときにLOOPとの間の実行を止めることです．

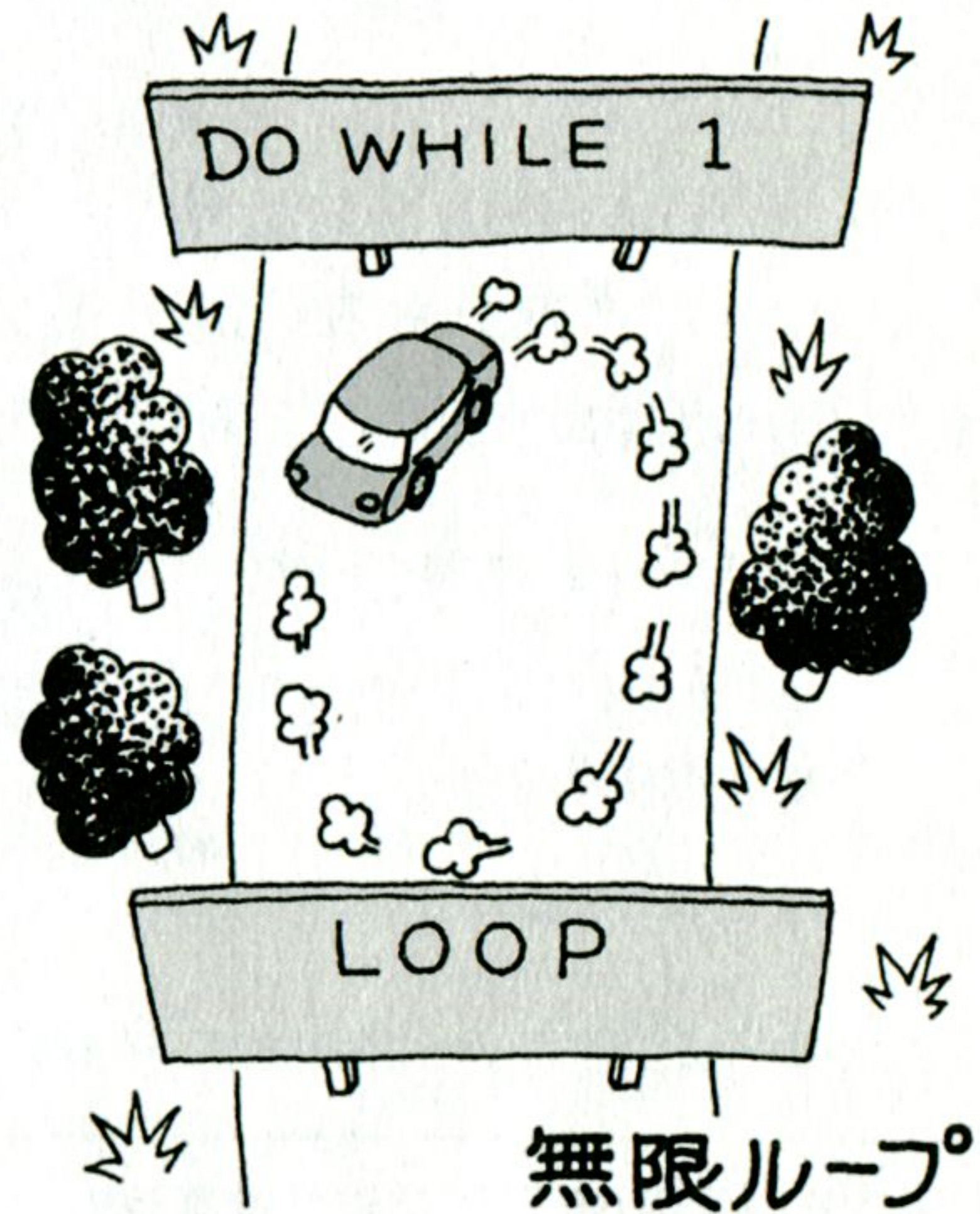
このように，DO UNTILとDO WHILEを使うと，何回繰り返すかわからない式を，ある条件になるまで繰り返し続けさせることができます．

あとに書く式を間違えると，無限ループに落ちいってしまうことがあるので，気を付けてください．

DO WHILE を使って，ある間をずっと繰り返して実行させることができます．ゲームとかシミュレーションでプログラムを1回やっておしまいということはまずありません．あるルーチンの間をずっと繰り返し回っていて，その中でキー入力により動いたり，宝を見付けたり，敵と戦ったりするのが普通です．このような場合，次のようにするのが普通です．

```
DO_WHILE_1
  :
  :
LOOP
```

これで，DO WHILE と LOOP の間を繰り返し行います．

DO と LOOP だけを使って，

```
DO
 :
 :
LOOP
```

のように書くこともできます．この場合，DO と LOOP の間を永久に繰り返すことになります．シミュレーションやゲームでよく使われるやり方です．

イグジット　ドゥ
EXIT DO

このままでは，2ヵ所の間を無限に回る無限ループになって，そこから出られなくなってしまいます．そこで，EXIT DO という命令があります．

```
C$_=_INKEY$
IF_C$_=_"Q"_THEN_EXIT_DO
```

のように，LOOP から抜け出す部分を中に入れておいてください．そうでないと，STOP キーを押す以外に止めることができなくなってしまいます．

★1，2，3，4…，トシチャン・メッケ

みんなが上手に隠れるようになると，なかなか見つかりません．これでは鬼もつまらないし，じっと隠れて待っている方もたいくつです．そこで，“カクレンボ”はさらに変身します．

つまり，数を数えて，ある一定の数まできたら，相手が隠れていようがいまいが探し始めるやり方です．時間が短いと，隠れる方も大忙がしで，なかなか楽しめました．

FOR NEXT（フォー ネクスト）

これがFOR NEXTの世界です．

FOR NEXTは，ある一連の動作を決められた回数繰り返すのに使います．“カクレンボ”にたとえるならば，次のようなプログラムのようになります．

```
FOR_I_=_1_TO_30
  Iを大声でいう
NEXT
探し始める
```

このように，1から30までを大声でいったあと，次の動作に移ります．

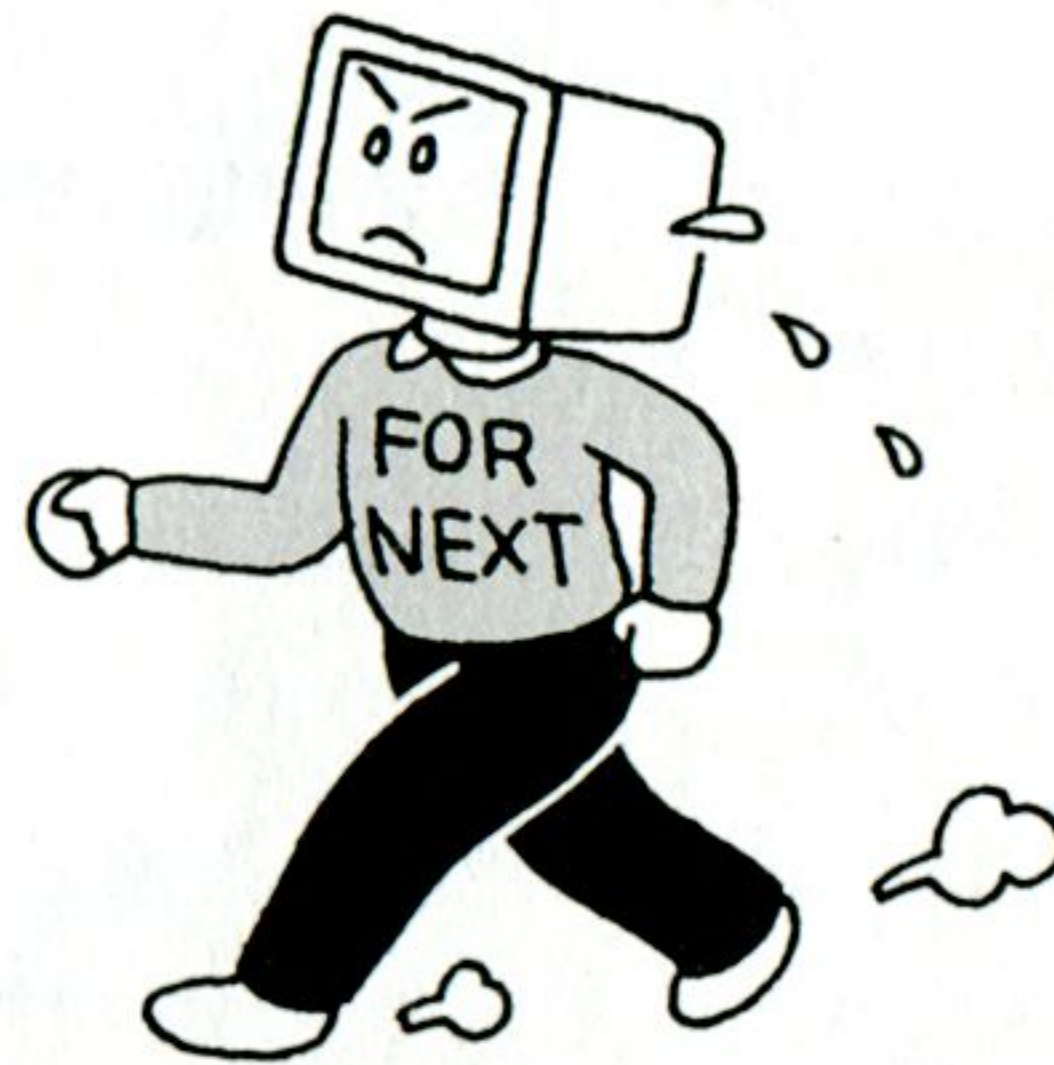
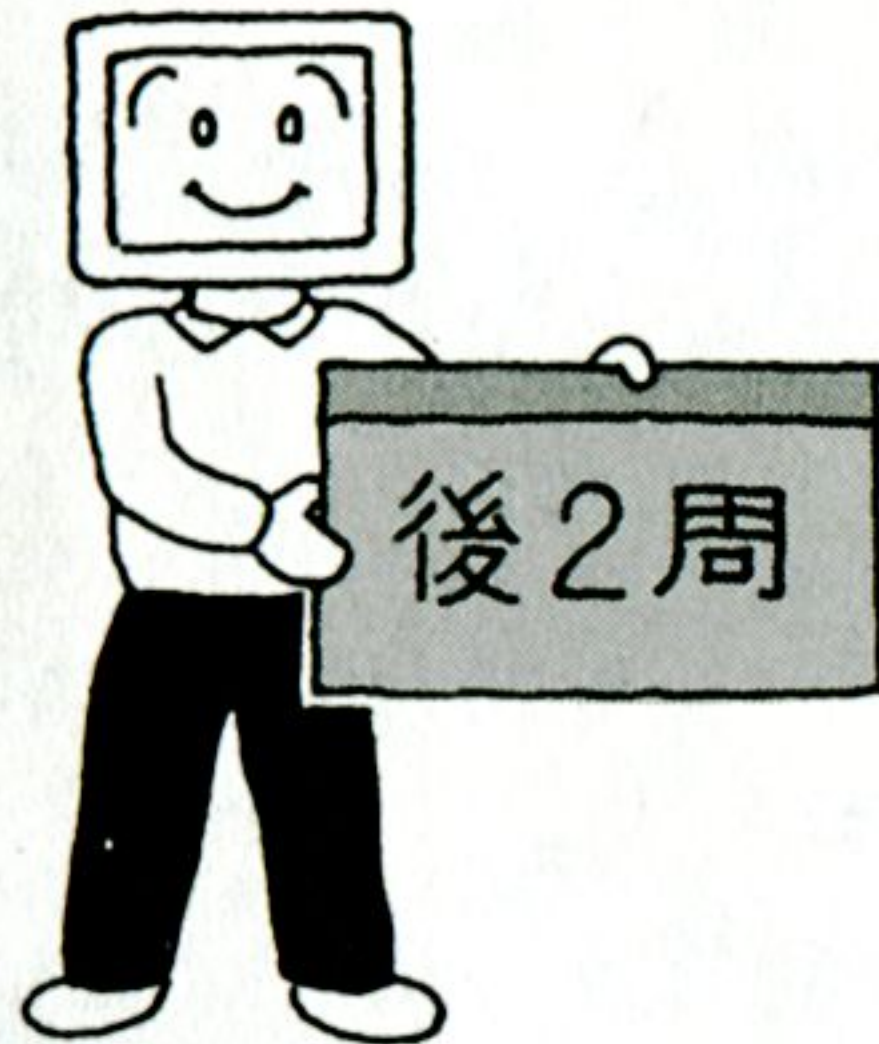
FORNEXT．BASプログラム

```
CLS
INPUT "100 ﾏﾃﾞ ﾉ ｽｳｼﾞｦ ｲﾚﾃ ｸﾀﾞｻｲ  ", N
GOUKEI = 0
KAZU = 0
FOR I = 0 TO N
    GOUKEI = GOUKEI + KAZU
    KAZU = KAZU + 1
NEXT
PRINT GOUKEI
END
```

FOR_I_=_0_TO_N　の式で，NEXTと書かれた式の間を0回目，1回目，2回目…と，N回まで繰り返します．あとになにも書かなければ，毎回の増分は1ですが，あとにSTEPと数を書き，毎回の増分を指定することもできます．

たとえば，

FOR_I_=_10_TO_20_STEP_2

とすると，10，12，14，…と２ずつ増やしていき，20になったところで止めます．これは，DO WHILE や DO UNTIL と違い，きちんと決めた数だけ繰り返したいときに使います．

FOR NEXT

Quick BASIC

モジュール使って効率良く！

SUB と FUNCTION

SUB(サブ) と FUNCTION(ファンクション) というと，すくにGOSUB(ゴーサブ)やDEFFN(デファインファンクション) が思い出されますが，ここでいう SUB と FUNCTION は，そのことではありません．

Quick BASIC では，SUB や FUNCTION を本体のプログラムと別個に作り，本体から必要に応じて呼び出せるようになっています．また，SUB や FUNCTION の中で使う変数は，本体の変数とまるっきり独立させることができます．これにより，1つ1つの SUB や FUNCTION をあたかも部品のように使うことができます．つまり，一度作っておいた部品を，必要に応じて他のプログラムにコピーして使うことができます．このことは，性能を十分に確かめたルーチンが使え，2度同じルーチンを書いたり，変数の名前を変えたりする手間が完全に省けます．さらにこうして作られた SUB や FUNCTION は，メモリ上で通常のコード部分と別の部分を使うので，コード部分のメモリを有効に使うことができます．

SUB と FUNCTION とも，原則的な使い方は良く似ているのですが，ルーチンで仕事を行ったあと，何か値を返して来るものが FUNCTION，値を返してこないものが SUB と考えてください．

メインのルーチンと SUB，または FUNCTION のルーチンの関係は，次のようになります．

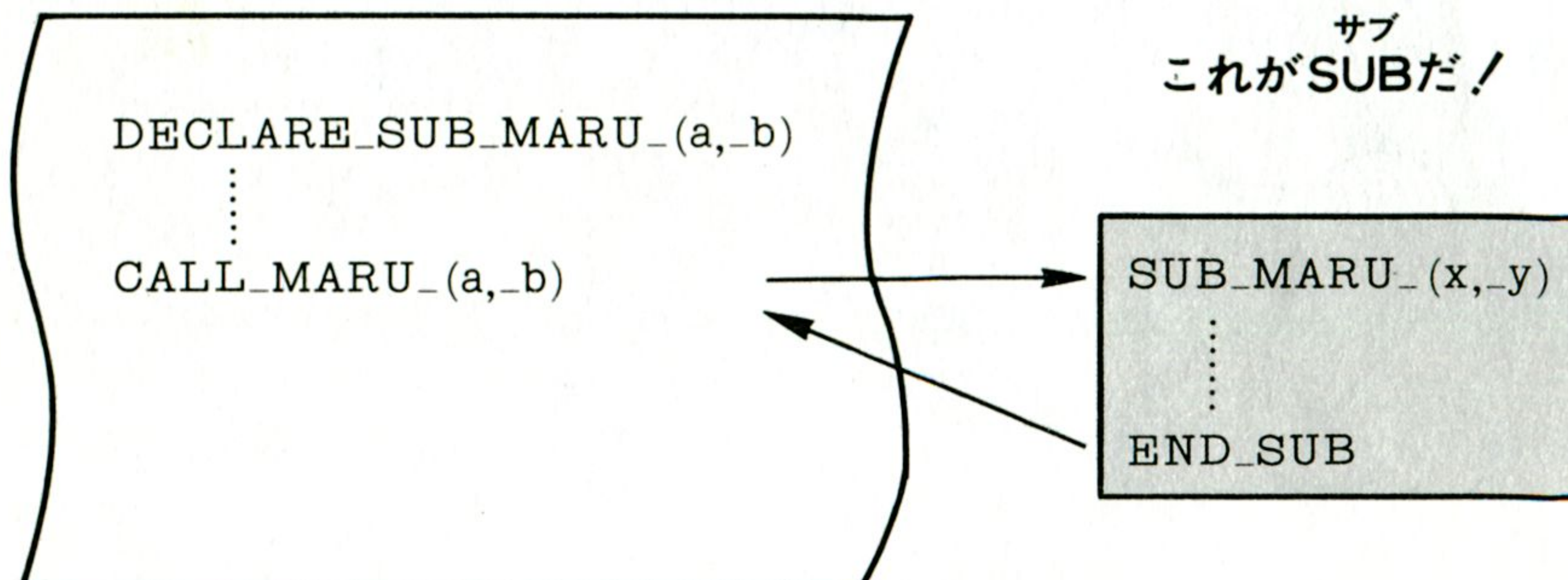
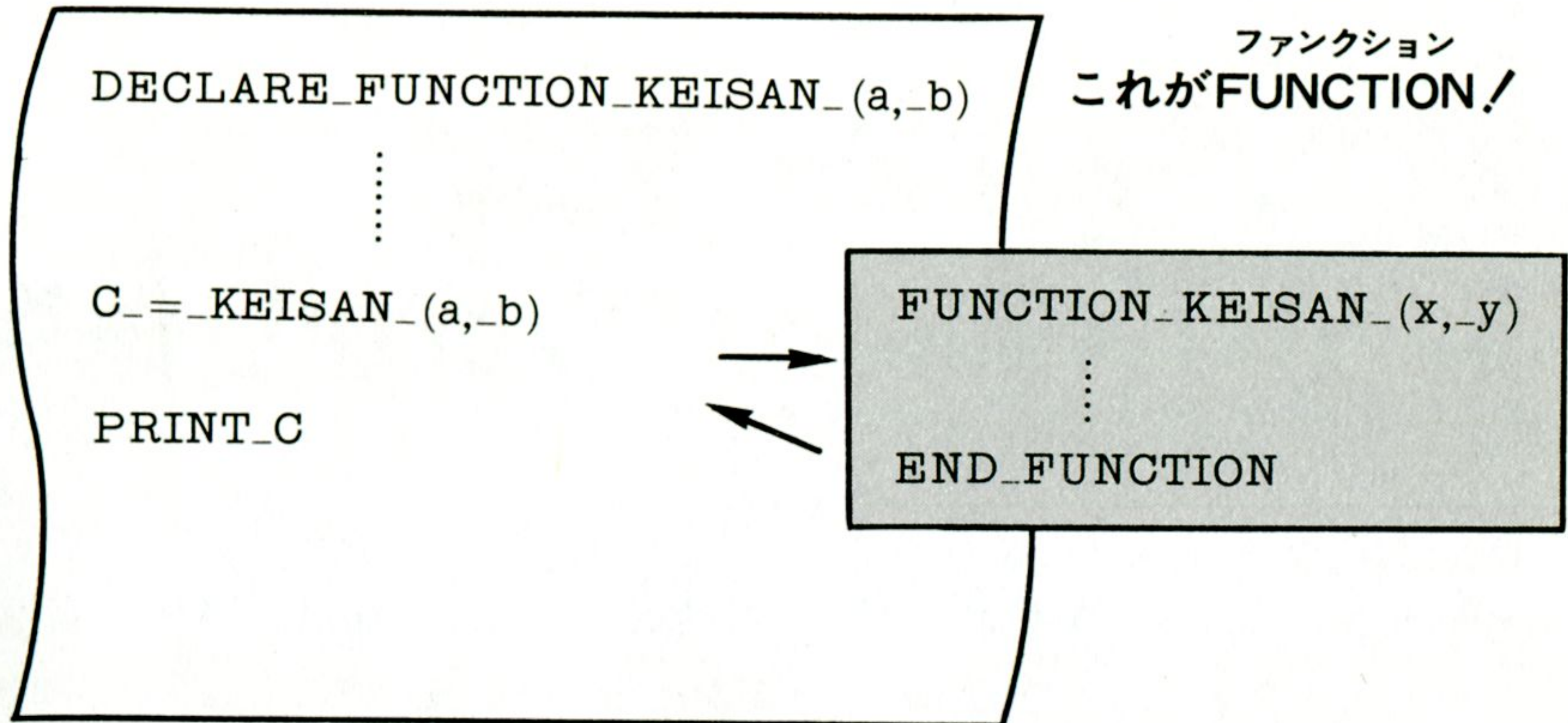
```
DECLARE_SUB_MARU_(a,_b)
    :
CALL_MARU_(a,_b)
```

これがSUB(サブ)だ！

```
SUB_MARU_(x,_y)
    :
END_SUB
```

```
DECLARE_FUNCTION_KEISAN_(a,_b)
        ⋮
C_=_KEISAN_(a,_b)

PRINT_C
```

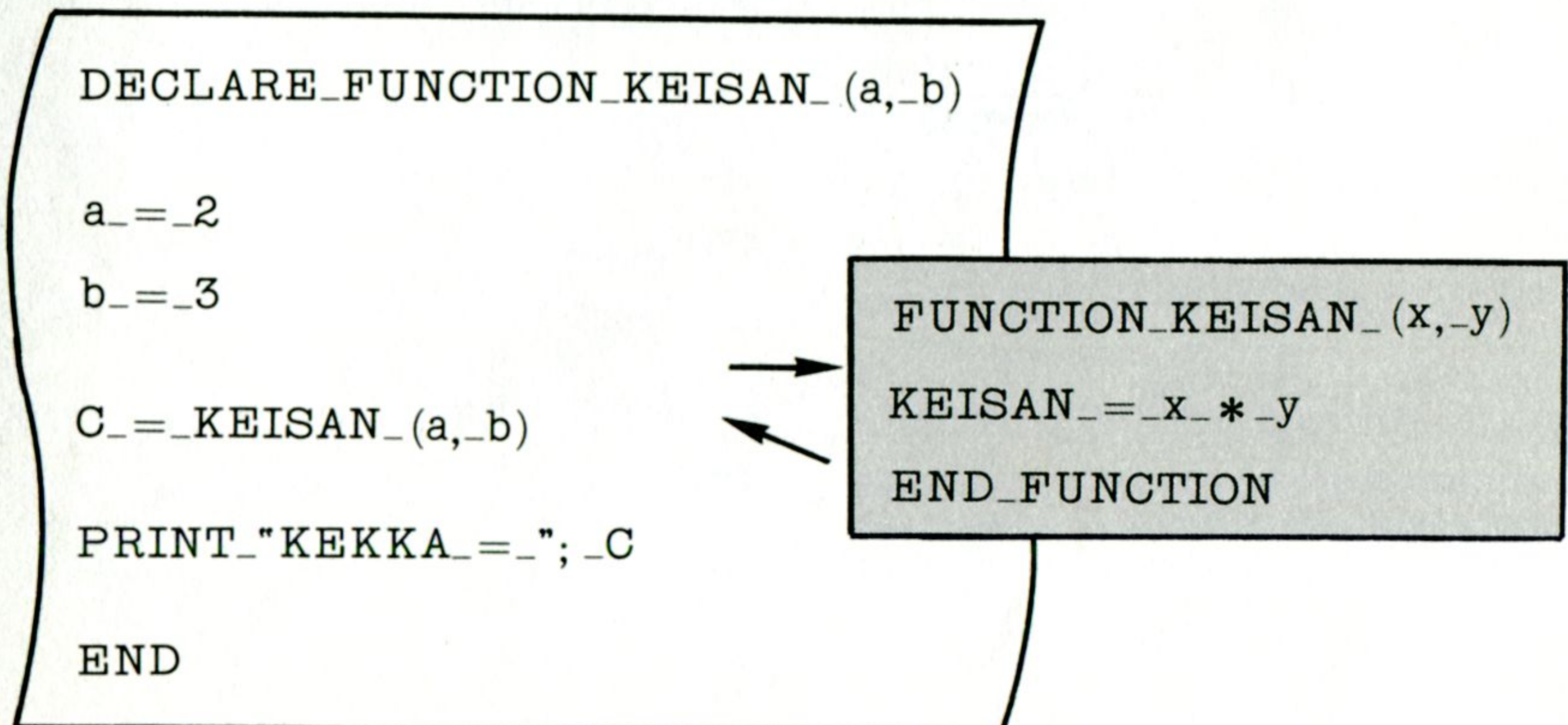
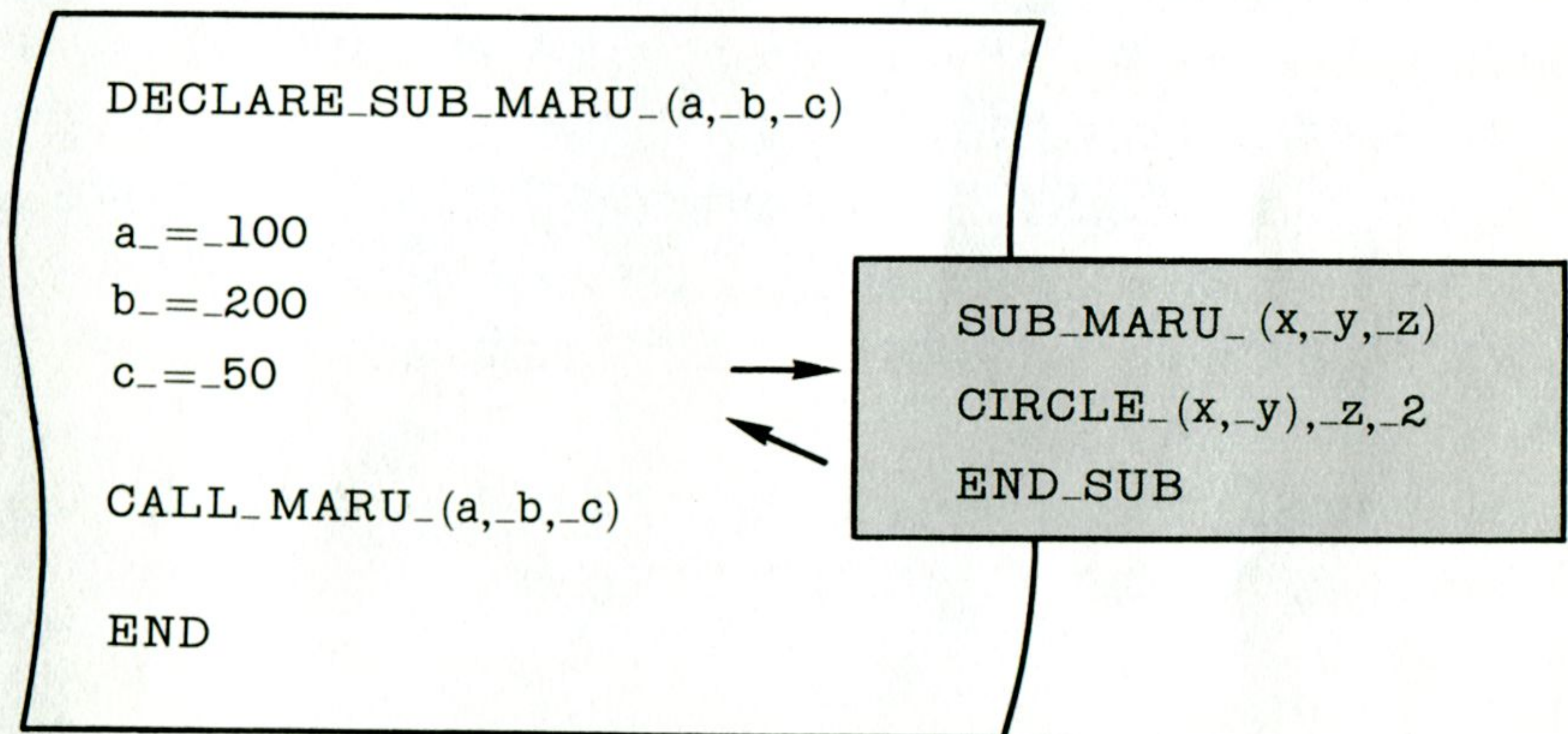


```
FUNCTION_KEISAN_(x,_y)
        ⋮
END_FUNCTION
```

SUB または FUNCTION での処理が終了すると，プログラムの制御は呼び出した次の行にもどってきます．

```
DECLARE_SUB_MARU_(a,_b,_c)

a_=_100
b_=_200
c_=_50

CALL_MARU_(a,_b,_c)

END
```

```
SUB_MARU_(x,_y,_z)
CIRCLE_(x,_y),_z,_2
END_SUB
```

```
DECLARE_FUNCTION_KEISAN_(a,_b)

a_=_2
b_=_3

C_=_KEISAN_(a,_b)

PRINT_"KEKKA_=_";_C

END
```

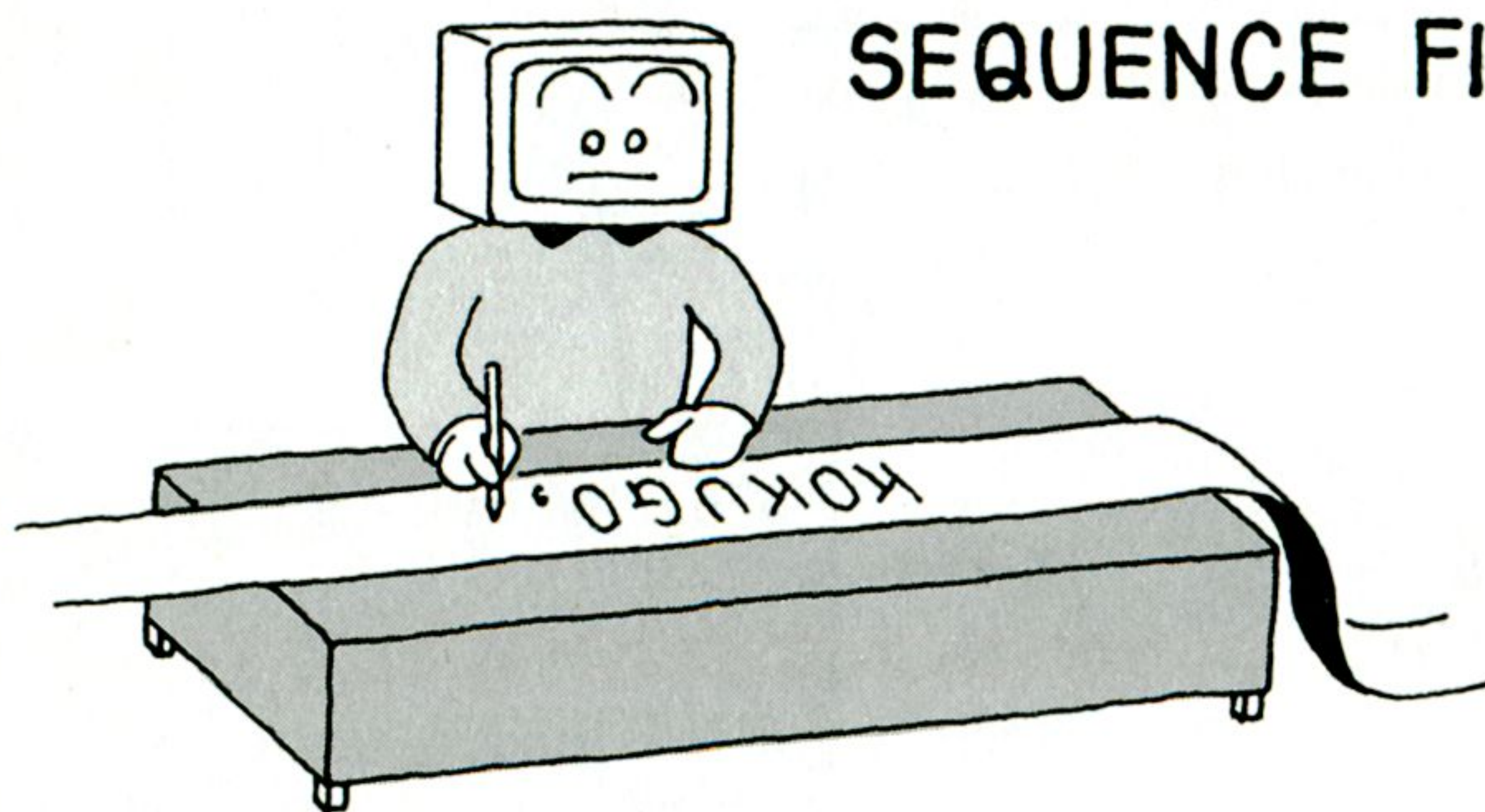
```
FUNCTION_KEISAN_(x,_y)
KEISAN_=_x_*_y
END_FUNCTION
```

Quick BASIC

大事なデータは，しっかり保存

Quick BASIC を使用していると，一時的にあるデータをディスクにセーブしておきたいときがあります．たとえば，プログラムの条件をときどき変えたいが，毎回エディタの中でデータを全部変えるのは大変だとか，１つのプログラムの結果を使って，次のプログラムの計算を行うが，毎回すべてのプログラムを実行すると余計な時間がかかるときなどです．

このような場合は，ファイルを作って必要な事項の保存を行うのが非常に有効です．ファイルを作ったあとは，パソコンの電源を切っても，しっかりディスクにセーブしてありますから安心です．

このようなファイルには，データを順番に保存，読出しして行くシーケンシャルファイルとデータに番号を付け，自由にそのどれかを選んで読み出せるランダムアクセスファイルの２種類があります．ここでは，シーケンシャルファイルについて見て行きましょう．

Quick BASIC でファイルを扱うときには，ファイルの名前ごとに番号を割り振り，実際の読み書きはこの番号に対して行います．

SEQUENCE FILEの書込み

KOKUGO,

次のSFILEOUT.BASプログラムを入力，保存，実行してみましょう．

```
SFILEOUT．BASプログラム
 CLS
 OPEN "SAMPLE" FOR OUTPUT AS #1
 WRITE #1, "KOKUGO"
 WRITE #1, 90
 WRITE #1, "SANSUU", 85
 CLOSE #1
 END
```

2行目の

OPEN_"SAMPLE"_FOR_OUTPUT_AS＃1

で，SAMPLEという名前のファイルを作り，そのファイルを書込み用のファイルとして使い，＃1という番号を与えています．BASICでは，同時に複数のファイルをオープンすることができます．

3行目からの

WRITE_＃1,_"KOKUGO"
WRITE_＃1,_90

で，＃1として選ばれたファイル，つまり，ここではSAMPLEというファイルにまずKOKUGOの文字，次にデータの値である90を書き出しています．そして次の行で

WRITE_＃1,_"SANSUU",_85

とあるところで先程のデータのあとにSANSUUの文字とSANSUUの値である85が付け加えられます．

このように新しいデータを書くたびにファイルの後ろ側に付け足されて行きます．

6行目の　CLOSE_＃1　ですが，この命令で先程までデータを書いてきたファイルを閉じています．

CLOSE_＃1　を書かないと，パソコンにはファイルの終わりが認識できないので，必ず忘れずに書いてください．

これでパソコンのディスクまたはハードディスク上にSAMPLEというファイルが作られ，その中には，必要な文字と数値が保存されました．

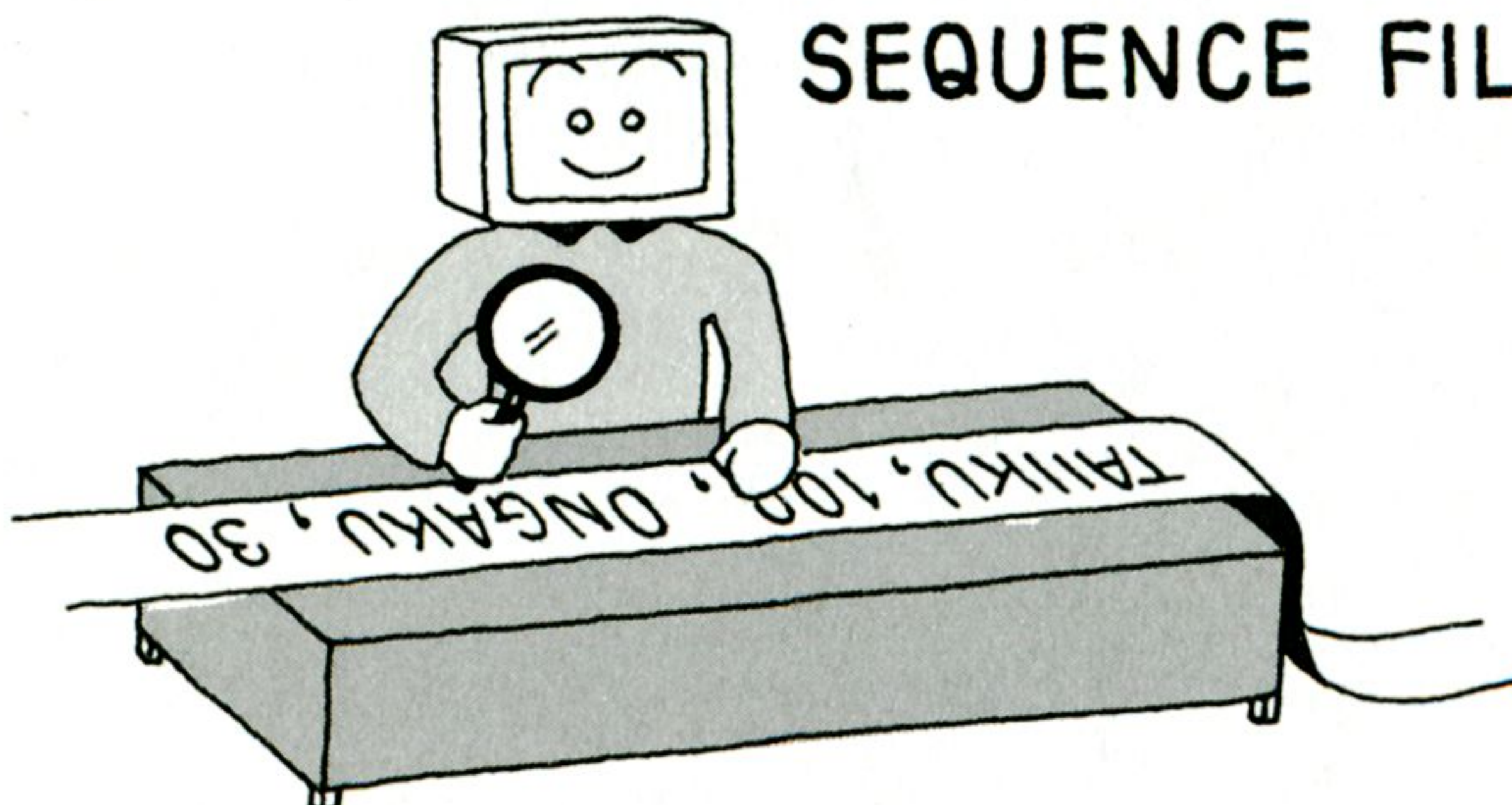
文字列を書き込む時には文字列の前後を”　　”(ダブルクォーテーションマーク)で囲んでください．

ファイルに書き込むプログラムについて勉強したので，今度は保存されているファイルからデータを取り込むプログラムについて見ていきましょう．

次の SFILEIN.BAS プログラムを入力，保存，実行してみてください．

ＳＦＩＬＥＩＮ．ＢＡＳプログラム

```
CLS
OPEN "SAMPLE" FOR INPUT AS #1
INPUT #1, A$
INPUT #1, TENSUUA
INPUT #1, B$
INPUT #1, TENSUUB
CLOSE #1
PRINT A$
PRINT TENSUUA
PRINT B$
PRINT TENSUUB
GOUKEI = TENSUUA + TENSUUB
PRINT "GOUKEI"
PRINT GOUKEI
HEIKIN = GOUKEI / 2
PRINT "HEIKIN"
PRINT HEIKIN
END
```

このプログラムでは，SAMPLE というファイルを呼び出し，先程記録した 2 つの文字列と数字を読み出して，その数字の合計と平均を求めています．

SEQUENCE FILEの読出し

先程と大きく違うのは，2行目のOPEN "SAMPLE"のあとがINPUT(インプット)となっていることです．これはSAMPLEというファイルを，そこからデータを読み出すのに使うということを示しています．

4行目の

INPUT_#1, TENSUUA

で，TENSUUAという変数の値をファイルの中に蓄わえられている数にしています．ここでは先程，このSAMPLEというファイルに記録した90という数値がTENSUUAという変数に入ります．

同じように，SAMPLEというファイルに記録した85という数値をTENSUUBという変数に取り込み，両者の合計と平均を出しています．

このプログラムのように，入力のときと出力のときで変数の名前が違っても構いません．また，数値の他に文字変数を記録することもできます．

ここで注意したいのは，変数の種類と順番です．文字変数で記録したら読み出すときも文字変数に，というように必ず記録したときの変数の種類と順番が同じになるように読み出してください．

通常の書込みのほかに，いままで作っておいたファイルに何か情報を付け加えたいときがあります．このときは，下のプログラムのようにファイルをオープンするときにAPPEND(アペンド)を使ってください．

```
SFILEAP．BASプログラム
 CLS
 OPEN "SAMPLE" FOR APPEND AS #1
 WRITE #1, "RIKA  "
 WRITE #1, 100
 WRITE #1, "SHAKAI", 80
 CLOSE #1
 END
```

コラム

8色しか出せない機種への対応

8色しか出せない機種をお持ちの方は，そのままでは動かないプログラムがいくつかあります．そのような機種をお持ちの方は，次のようにプログラムを書き換えてください．

1．すべての絵をかくプログラム中の SCREEN_88, のあとの数字を0に替えてください(機種によっては，SCREEN_88, のあとの数字を入れずに SCREEN_88,_, のようにブランクでも構いません)．

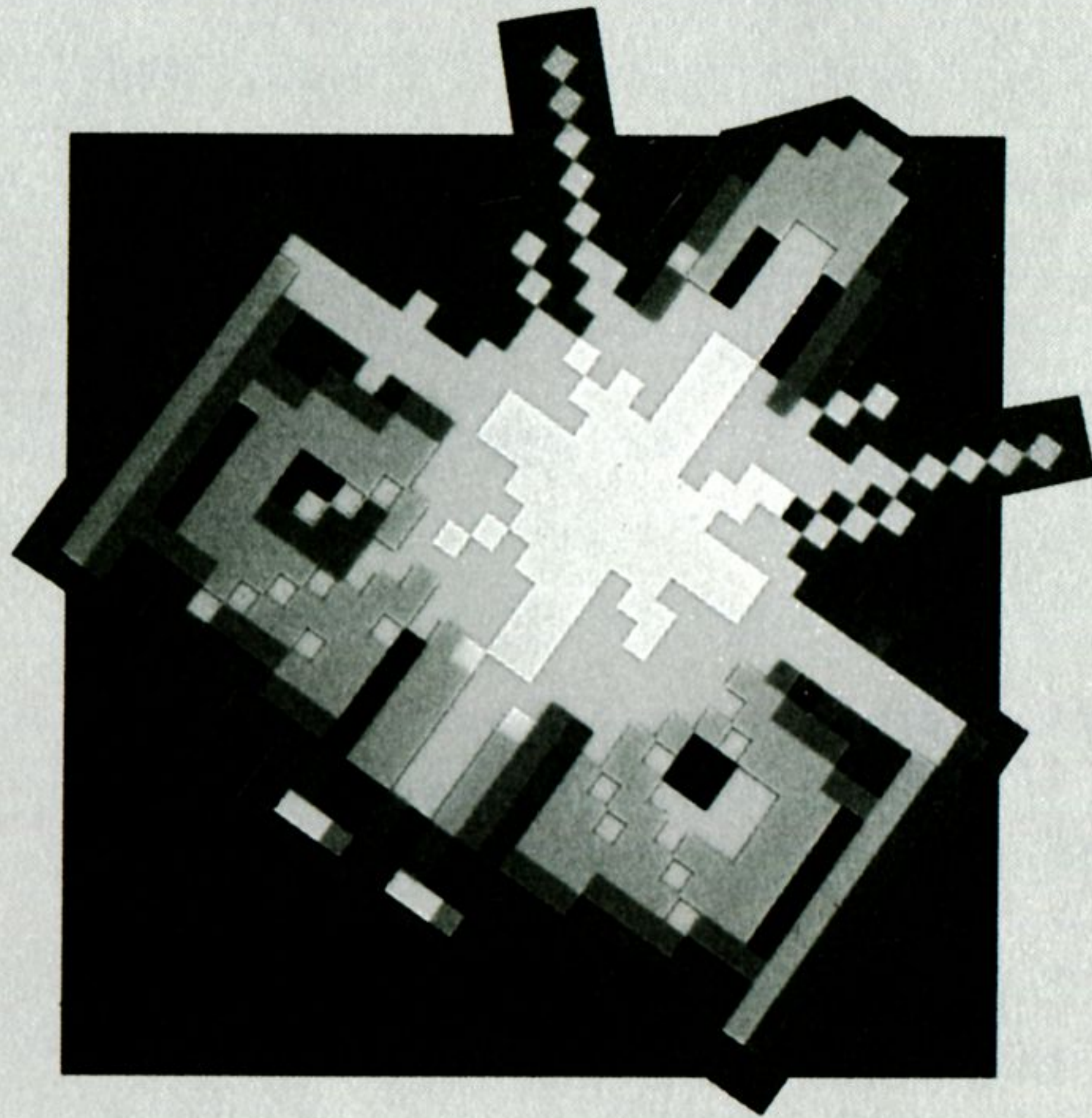
2．100ページに掲載してあるキャラクタジェネレータ CGENE.BAS の1行目 SCREEN 88, 3, 1, 1 の3を0としてください．

次に，同じ CGENE.BAS の4行目 WAKUCOLOR＝12 の12を3に替えてください(とくに3でなくても，自分でキャラクタに使わないと思う1~7までの色番号なら何でも構いません)．

次に，同じ行の MAXCOLOR＝15 の15を7に替えてください．これらの改造で，CGENE.BAS が8色対応となります．

いまや，大半のソフトウェアが16色対応または16色専用です．専用ソフトの中には8色機では動かないものがあります．VM/VF などの機種には，サードパーティから対応する16色カラーボードが安価で販売されています．このボードとアナログ RGB モニタを使うことにより，8色機でも16色が使えるようになります．

いよいよ本格的ゲームの製作に入ろう！

Quick BASIC

自分自身のスペシャルゲームを作ろう！

では，いよいよゲーム作りに取りかかりましょう．

これから作るのは，世界にたった1つしかない，あなたオリジナルのスペシャルゲームです．はりきって挑戦してください．

★まず画面の絵を動かしてみよう

いままでの部分では，動きがまったくありませんでした．動きがなければ全然おもしろくありません．

ここでは，画面の絵を動かすことから始めましょう．

★箱を動かす

とりあえず，まず動きの基本を説明するために，画面に箱をかき，それを動かしてみることにしましょう．

次のプログラムを入力，保存したあと実行してみてください．

```
ＨＡＫＯ．ＢＡＳプログラム
 CLS
 TATE = 200
 YOKO = 340
 LINE (YOKO, TATE)-(YOKO + 20, TATE + 20), 7, B
 DO WHILE 1
 BEGIN:
 A$ = INKEY$
 IF A$ = "" THEN GOTO BEGIN
     IF A$ = "2" THEN TATE = TATE + 20
     IF A$ = "4" THEN YOKO = YOKO - 20
     IF A$ = "6" THEN YOKO = YOKO + 20
     IF A$ = "8" THEN TATE = TATE - 20
     IF A$ = "Q" THEN END
     LINE (YOKO, TATE)-(YOKO + 20, TATE + 20), 7, B
 LOOP
```

プログラムをスタートさせると，画面の中央に箱が現れます．ここで，数字の4のキーを押すと，箱の左にもう1つ箱が現れます．6キーを押すと右に，2キーを押すと下に，8キーを押すと上に動きます．プログラムを止めたくなった場合は，Qキーを押すと止まります．

プログラムを見てみましょう．

まず，TATE_=_200　YOKO_=_340　縦と横の初期値を設定しています．次のLINE文で最初の箱を描いています．そして，DO_WHILE_1　で，強制的に無限ループを作り出し，LOOPまでの間を繰り返しています．

このプログラムで大事な点は，7行目の　A$_=_INKEY$（インキーダラー）　と，あとの行のIF_A$_="2"　で始まる一連の行です．

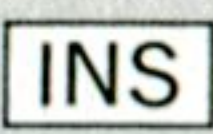
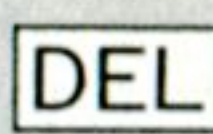
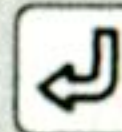
まず，INKEY$は，キーボードのどのキーが押されたかを1文字単位でプログラムに取り込みます．　A$_=_INKEY$　と書くと，A$がそのとき押されたキーに対応する文字となります．

次に，いま入力されたA$がどの文字なのかを，1文字ごとに確認して行きます．このとき，調べたい文字を必ず"（ダブルクォーテーションマーク）で前後をはさんでください．IF文ですので，押されたキーが"　"の中の文字と一致したら，IF文のあとの式を実行します．

INKEY$で取り込まれた文字は，大文字，小文字の区別があります．もし，キーボードのCAPSロックされていてもいなくても，同じように動作させたい場合は，次のように大文字，小文字，1つずつに対応する式を作っておけばOKです．

```
IF_A$_=_"Q"_THEN……
IF_A$_=_"q"_THEN……
```

超完璧主義者の人は，カナキーのめんどうも見て，

```
IF_A$_=_"タ"_THEN……
```

まで書いておけばパーフェクトです．

コラム

特殊キーの使用(1)

数字キーでの動きの入力は，作る方にとっては簡単ですが，やはり，矢印キー[↑]やインサートキー[INS]，デリートキー[DEL]，リターンキー[↵]，などが使えた方が便利です．また，プログラムによってはぜひ必要になることもあるでしょう．そこで，特殊キーでの入力を見てみましょう．

次のプログラムを見てください．

TOKUSHU．BASプログラム

```
CLS
PRINT "PUSH THE ARROW KEY "
HAJIME:
C$ = INKEY$
IF C$ = "" THEN GOTO HAJIME
KEY1 = ASC(LEFT$(C$, 1))
KEY2 = ASC(RIGHT$(C$, 1))
IF KEY1 = 0 AND KEY2 = 75 THEN PRINT "LEFT "
IF KEY1 = 0 AND KEY2 = 77 THEN PRINT "RIGHT"
IF KEY1 = 0 AND KEY2 = 72 THEN PRINT " UP  "
IF KEY1 = 0 AND KEY2 = 80 THEN PRINT "DOWN "
GOTO HAJIME
```

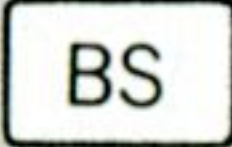
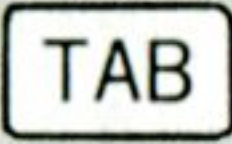
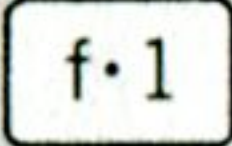
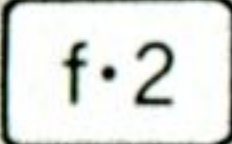
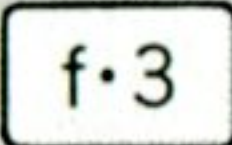
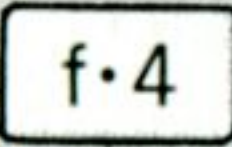
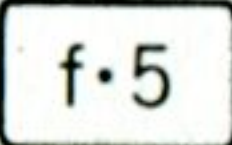
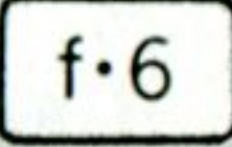
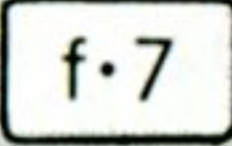
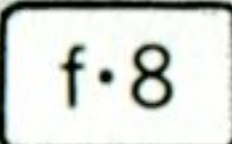
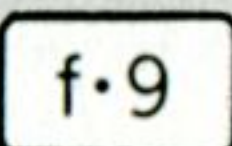
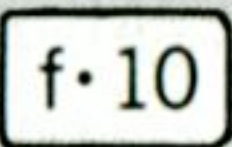
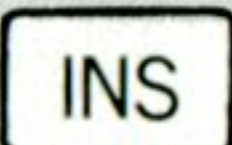
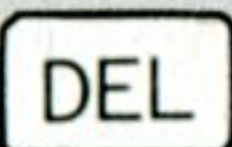
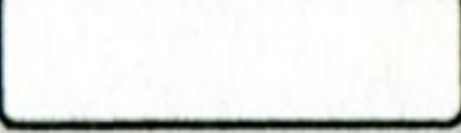
キー入力を行ったとき，1文字として入力されるものと2文字分として扱われるキーと2種類あります．これをASC(アスキー)関数を使ってアスキー成分を取り出し，その値があらかじめ調べておいたキーの対応表と一致するかどうかを調べます．

C$_=_INKEY$　で，キー入力した文字列をC$というストリングに取り込みます．

KEY1_=_ASC(LEFT$(C$,_1))

で読み込んだC$ストリングを左から1文字分だけ読み，それのASCII(アスキー)コードを取り出します．

KEY2_=_ASC(RIGHT$(C$,_1))

でC$ストリングを右から1文字取り，そのASCIIコードを取り出しています．

次に，
IF_KEY1_=_0_AND_KEY2_=_75
THEN…などの文で，両方のASCIIコードが望むキーのものかどうかを確かめ，両方一致した時のみ所望の処理を行うようにしています．

特殊キーのリスト

キー		
⏎	リターンキー	KEY1_=_13
BS	バックスペース	KEY1_=_8
ESC	エスケープ	KEY1_=_27
TAB	タブ	KEY1_=_9
f･1	KEY1_=_0	KEY2_=_59
f･2	KEY1_=_0	KEY2_=_60
f･3	KEY1_=_0	KEY2_=_61
f･4	KEY1_=_0	KEY2_=_62
f･5	KEY1_=_0	KEY2_=_63
f･6	KEY1_=_0	KEY2_=_64
f･7	KEY1_=_0	KEY2_=_65
f･8	KEY1_=_0	KEY2_=_66
f･9	KEY1_=_0	KEY2_=_67
f･10	KEY1_=_0	KEY2_=_68
INS	KEY1_=_0	KEY2_=_82
DEL	KEY1_=_0	KEY2_=_83
ROLL UP	KEY1_=_0	KEY2_=_81
ROLL DOWN	KEY1_=_0	KEY2_=_73
（スペースキー）	スペース	KEY1_=_32
←	KEY1_=_0	KEY2_=_75
→	KEY1_=_0	KEY2_=_77
↑	KEY1_=_0	KEY2_=_72
↓	KEY1_=_0	KEY2_=_80

いまのプログラムでは，キーを押すごとに箱が増え，たくさんの箱が画面に残ってしまいます．

絵をかくときには，前の絵を消してからかかないと，画面に残骸がいっぱい残ってしまいます．そこで，次のプログラムを入力して実行してみてください．前の HAKO.BAS を読み出して修正すれば簡単です．

OLDHAKO．BASプログラム

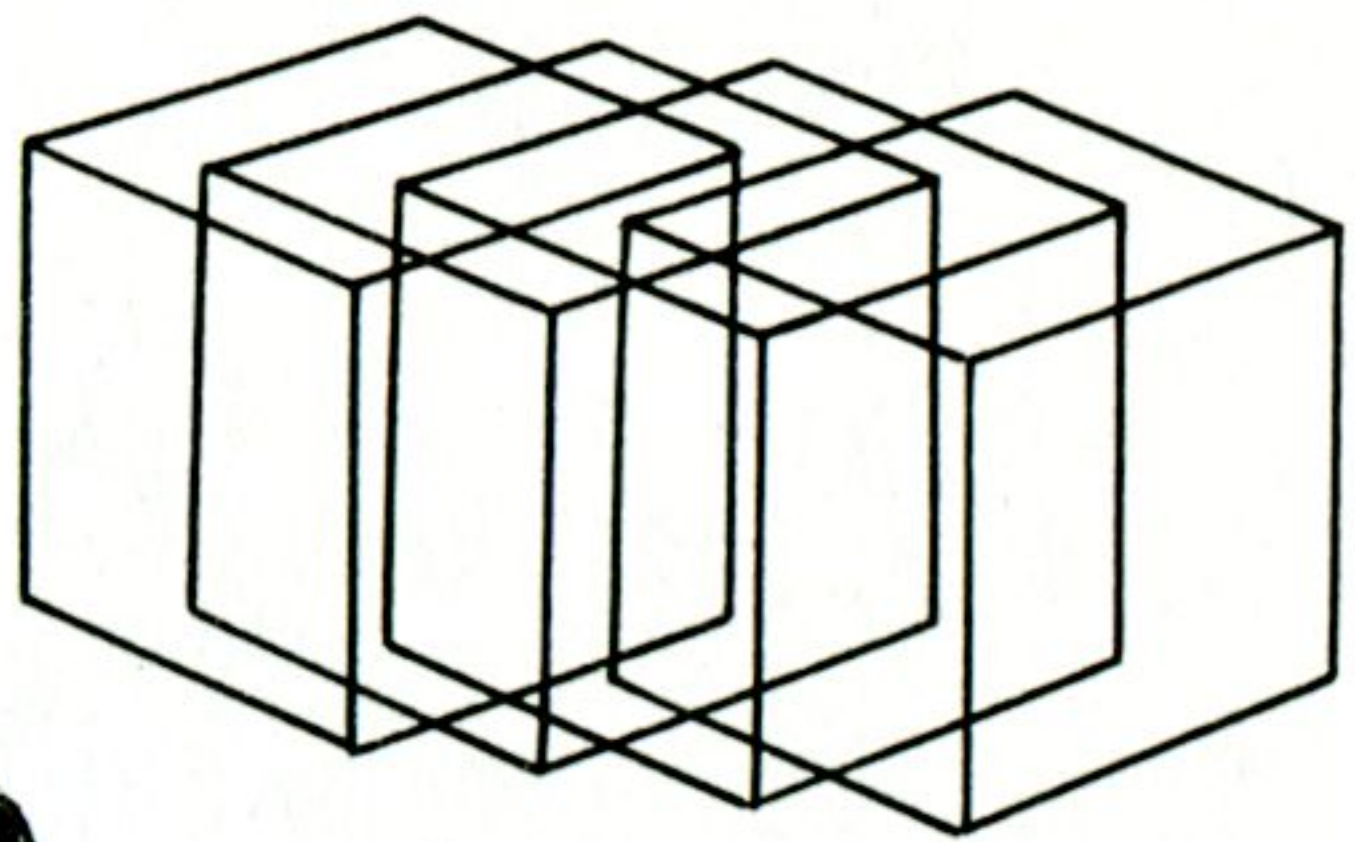
```
CLS
TATE = 200
YOKO = 340
OLDTATE = TATE
OLDYOKO = YOKO
LINE (YOKO, TATE)-(YOKO + 20, TATE + 20), 7, B
DO WHILE 1
BEGIN:
A$ = INKEY$
IF A$ = "" THEN GOTO BEGIN
    IF A$ = "2" THEN TATE = TATE + 20
    IF A$ = "4" THEN YOKO = YOKO - 20
    IF A$ = "6" THEN YOKO = YOKO + 20
    IF A$ = "8" THEN TATE = TATE - 20
    IF A$ = "Q" THEN END
    LINE (OLDYOKO, OLDTATE)-(OLDYOKO + 20, OLDTATE + 20), 0, B
    LINE (YOKO, TATE)-(YOKO + 20, TATE + 20), 7, B
    OLDYOKO = YOKO: OLDTATE = TATE
LOOP
```

プログラムの16行目で箱をかく直前に，前にかいた箱の上に黒で線を引いて，前にかいた箱を消しています．

プログラムの18行目で，そのときかいた箱の位置，TATE と YOKO を別の変数 OLDTATE と OLDYOKO に記憶させています．これは，次の回で絵を消すときに，どこの部分を消したら良いかを記憶させるために行っています．

絵を消してから次の絵をかくまでに時間があると，画面上で絵がチラつきます．絵を消すのは，次の絵をかく直前で行いましょう．

いつまでも箱ばかり動かしていてもつまりません．次のパートでは，自分の好きなキャラクタを作れるキャラクタジェネレータに挑戦してみましょう．

アリャ！
何だか
わからないゾ!!

コラム

特殊キーの使用(2)

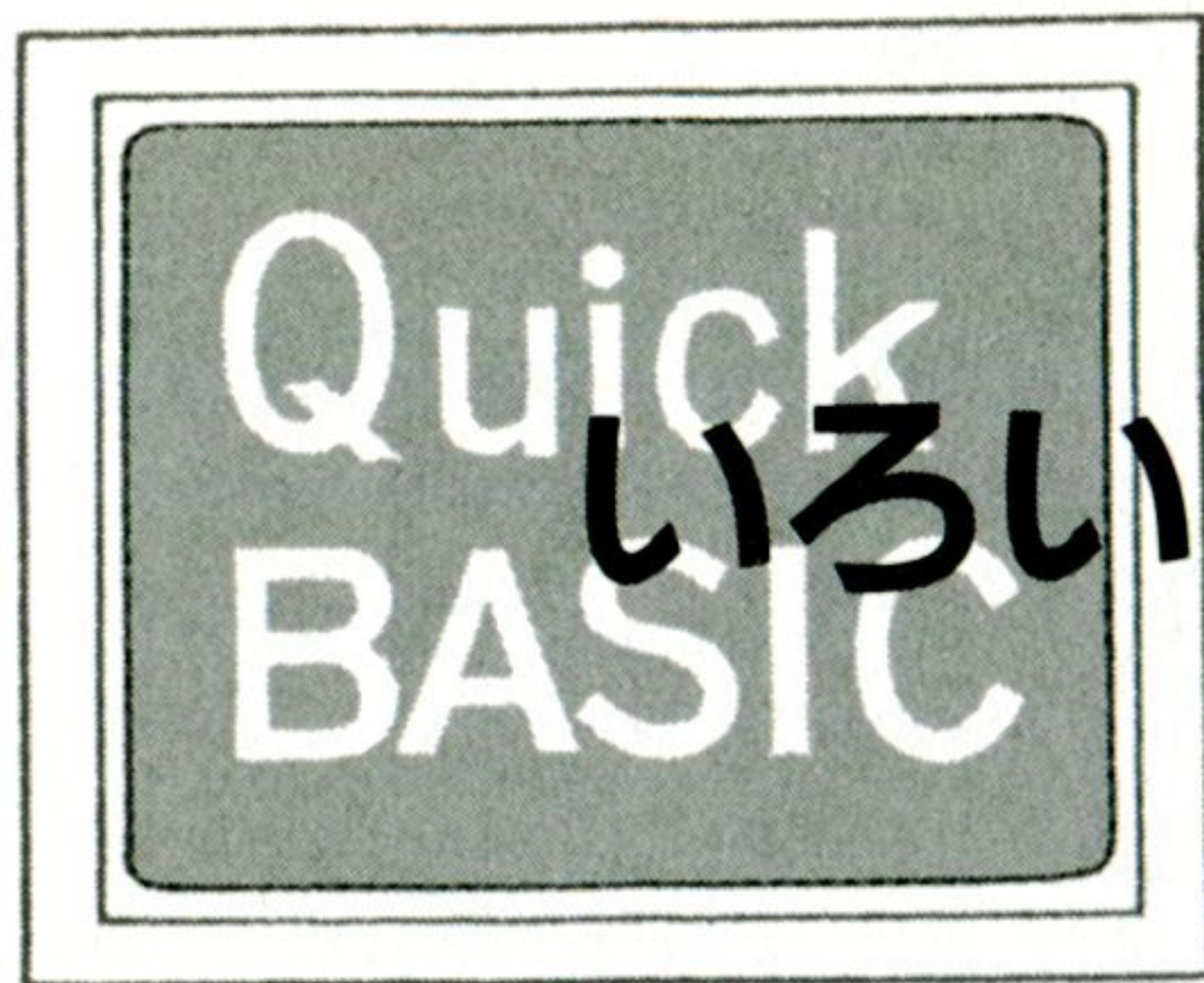
各キーに対応したKEY1とKEY2を得るためのプログラムKEYCHECK.BASを掲載させておきます．

これにより，シフトキー，コントロールキーとの組み合わせも含めて，必要なキーに対応するコードを得てください．

KEYCHECK．BASプログラム

```
CLS
50
PRINT "INPUT KEY CONBINATION"
100 A$ = INKEY$
IF A$ = "" THEN GOTO 100
C = ASC(LEFT$(A$, 1))
D = ASC(RIGHT$(A$, 1))
PRINT "KEY1="; C:
IF LEN(A$) > 1 THEN PRINT "KEY2="; D
GOTO 50
END
```

このKEYCHECK.BASのプログラムはすべてのキーのチェックを行うようにするため，特定のキーで止まるようには作っていません．止めるときはSTOPキーを押してください．また，STOPキー，COPYキー，CTRL+Cなど，いくつかのキーは最初からINKEY$では読めません．

Quick BASIC

いろいろ作ってしまおう！

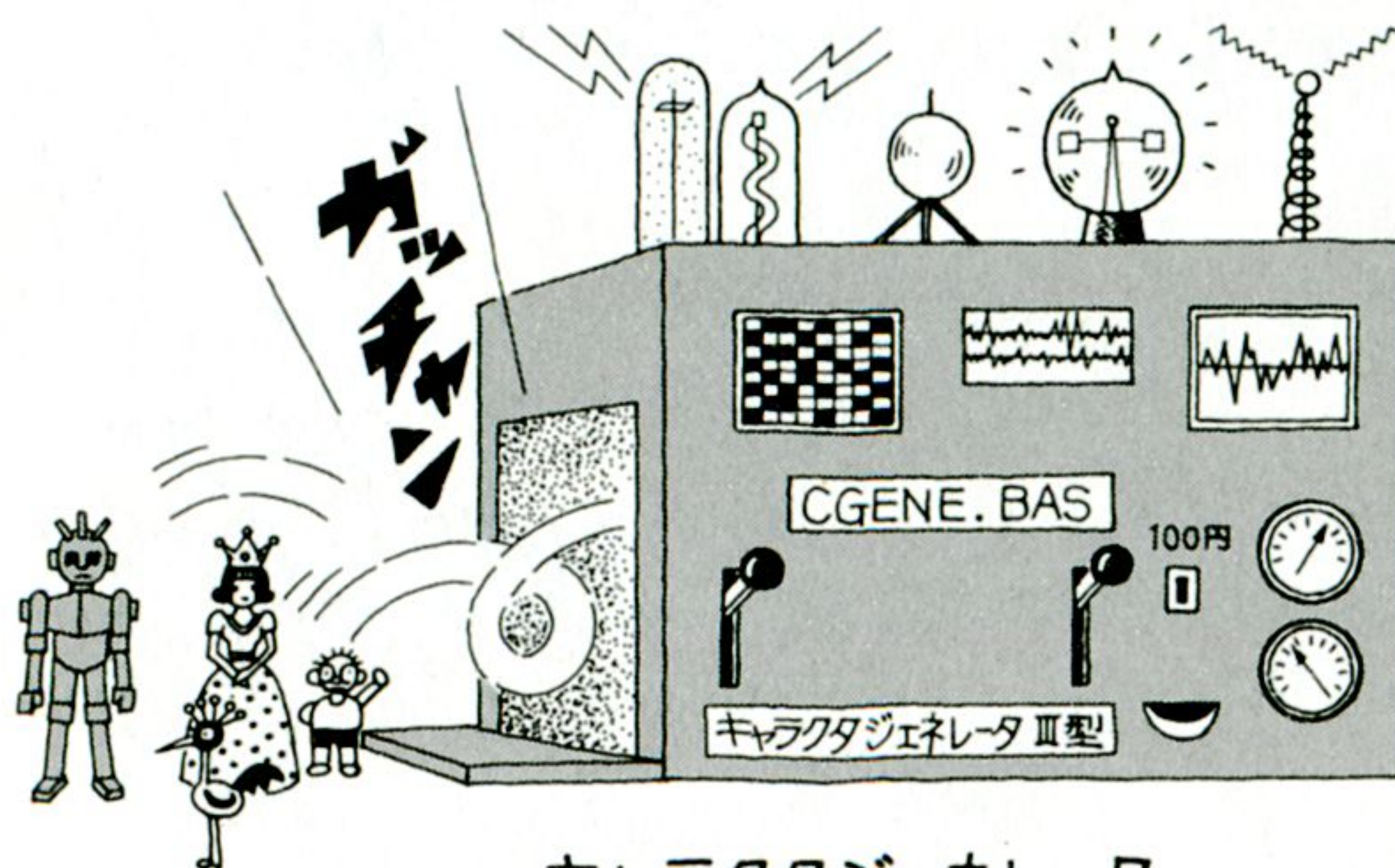
★かいじゅう，モビルスーツ，お姫様製造工場

ゲームの楽しさは，さまざまなキャラクタが画面狭しと動き回り，さまざまな冒険，アクションを行うところにあります．良くできたゲームになると，まるで自分がゲームの中の主人公になったような気分になります．ここで主人公に感情移入ができるかどうかの重要なポイントの1つとして，画面に出てくるキャラクタの姿があります．

ブタのように太ったお姫様や，見るからにか細いモビルスーツでは，ゲームの楽しみが半減してしまいます．

ここで，自分の思いどおりのキャラクタが作れたらどうでしょうか．ゲームの楽しさもまた倍加するものと思います．ここでは，キャラクタの作り方について述べて行きたいと思います．

画面に，自分の好きなロボットやモビルスーツまたはお姫様など，いろいろな形のものが登場したら楽しいですね．

画面に自分の好きな形をかく方法には，いろいろな方法があります．まず，その第一はPSET命令を使って，1点1点自分の望む点に色を塗って行く方法です．

キャラクタジェネレータ

KATACHIA．BASプログラム

```
SCREEN 88, 3, 1, 1
CLS
WINDOW SCREEN (0, 0)-(639, 399)
PSET (100, 100), 7
PSET (101, 100), 7
PSET (99, 101), 7
PSET (100, 101), 7
PSET (101, 101), 7
PSET (102, 101), 7
PSET (98, 102), 7
PSET (99, 102), 7
PSET (100, 102), 7
PSET (101, 102), 7
PSET (102, 102), 7
PSET (103, 102), 7
END
```

このやり方では絵をかいたり，形を替えるのは大変です．

次に，DATA（データ）文を使ったやり方があります．これは，決められた数の点の色だけをDATAとして持っておき，順番にセットしていくやり方です．この方法でも，絵をかくのは大変です．

KATACHIB．BASプログラム

```
SCREEN 88, 3, 1, 1
CLS
WINDOW SCREEN (0, 0)-(639, 399)
FOR I = 1 TO 6
READ A: READ B: READ C
PSET (A, B), C
NEXT
DATA 100,100,7,101,100,7,99,101,7,100,101,7,101,101,7,102,101,7
END
```

そこで，キャラクタジェネレータの登場です．

このキャラクタジェネレータを使えば，どんな絵でも簡単にかくことができます．また今回作った絵を保存しておき，色や形を変えて別の形を作るのも簡単です．

次のプログラムを間違えないように，ていねいに入力してください．

CGENE.BASプログラム

```
 SCREEN 88, 3, 1, 1
 WINDOW SCREEN (0, 0)-(639, 399)
 CLS
 COL = 7: WAKUCOLOR = 12: MAXCOLOR = 15
 WIDTH 80, 25
  A = 0: B = 0
  INSERT = 0
  DIM F%(3000)
  GET (500, 100)-(532, 132), F%
  DIM K%(300)
  GET (11 + B * 12, 11 + A * 12)-(21 + B * 12, 21 + A * 12), K%
  DIM L%(200)
  GET (22, 32)-(38, 48), L%
  PAINT (50, 50), 115
  DIM X%(200)
  GET (10, 10)-(15, 15), X%
  DIM Z%(3000)
  DIM KZ%(3000)
 11110 CLS
 10
    Y = 32
 20
    Z = 32
  CLS
  LINE (610, 30)-(630, 50), WAKUCOLOR, B
  PAINT (620, 40), COL, WAKUCOLOR
  FOR I = 0 TO MAXCOLOR
  LINE (610, 80 + 20 * I)-(630, 100 + 20 * I), WAKUCOLOR, B
  NEXT I
  FOR I = 0 TO MAXCOLOR
  PAINT (620, 90 + 20 * I), I, WAKUCOLOR
  NEXT I
  FOR I = 0 TO Y
  LINE (10, 10 + I * 12)-(10 + Z * 12, 10 + I * 12), WAKUCOLOR
  NEXT I
  FOR J = 0 TO Z
  LINE (10 + J * 12, 10)-(10 + J * 12, 10 + Y * 12), WAKUCOLOR
  NEXT J
  PUT (12, 12), X%
  PUT (600, 90 + COL * 20), X%
 LOCATE 2, 70: PRINT "OFF "
 LOCATE 17, 61: PRINT "L=LOAD"
 LOCATE 18, 61: PRINT "S=SAVE"
 LOCATE 19, 61: PRINT "Q=QUIT"
 LOCATE 20, 61: PRINT "0=PAINT"
 LOCATE 21, 61: PRINT "8=UP"
```

```
LOCATE 22, 61: PRINT "2=DOWN"
LOCATE 23, 61: PRINT "4=LEFT"
LOCATE 24, 61: PRINT "6=RIGHT"
LOCATE 3, 58: PRINT "TATE="; A + 1
LOCATE 3, 68: PRINT "YOKO="; B + 1
LINE (15 + B * 12, 2)-(18 + B * 12, 8), 7, BF
LINE (398, 15 + A * 12)-(404, 18 + A * 12), 7, BF
100 C$ = INKEY$
IF C$ = "" THEN GOTO 100
KEY1 = ASC(LEFT$(C$, 1))
KEY2 = ASC(RIGHT$(C$, 1))
   IF KEY1 = 0 AND KEY2 = 82 THEN INSERT = 1
   IF KEY1 = 0 AND KEY2 = 83 THEN INSERT = 0
   IF C$ = "L" THEN GOTO 4000
   IF C$ = "1" THEN A = A + 1: B = B - 1
   IF C$ = "2" THEN A = A + 1
   IF C$ = "3" THEN A = A + 1: B = B + 1
   IF C$ = "4" THEN B = B - 1
   IF C$ = "6" THEN B = B + 1
   IF C$ = "7" THEN A = A - 1: B = B - 1
   IF C$ = "8" THEN A = A - 1
   IF C$ = "9" THEN A = A - 1: B = B + 1


   IF KEY1 = 0 AND KEY2 = 75 THEN B = B - 1
   IF KEY1 = 0 AND KEY2 = 77 THEN B = B + 1
   IF KEY1 = 0 AND KEY2 = 72 THEN A = A - 1
   IF KEY1 = 0 AND KEY2 = 80 THEN A = A + 1
   IF C$ = "Q" THEN GOSUB OWARI
   IF C$ = "q" THEN GOSUB OWARI
   IF C$ = "Z" THEN GOSUB RESTART
   IF A < 0 THEN A = 0
   IF A > Z - 1 THEN A = Z - 1
   IF B < 0 THEN B = 0
   IF B > Y - 1 THEN B = Y - 1
PUT (11 + EB * 12, 11 + EA * 12), K%, PSET
GET (11 + B * 12, 11 + A * 12)-(21 + B * 12, 21 + A * 12), K%
MM2 = 4
IF POINT(12 + B * 12, 12 + A * 12) = 4 THEN MM2 = 15
LINE (11 + B * 12, 11 + A * 12)-(21 + B * 12, 21 + A * 12), (MM2), BF
LOCATE 3, 58: PRINT "TATE="; A + 1
LOCATE 3, 68: PRINT "YOKO="; B + 1
LINE (15 + EB * 12, 2)-(18 + EB * 12, 8), 0, BF
LINE (15 + B * 12, 2)-(18 + B * 12, 8), 7, BF
LINE (398, 15 + EA * 12)-(404, 18 + EA * 12), 0, BF
LINE (398, 15 + A * 12)-(404, 18 + A * 12), 7, BF
IF INSERT = 1 THEN
PUT (12 + B * 12, 12 + A * 12), X%, PSET
```

```
: PAINT (13 + B * 12, 13 + A * 12), COL, WAKUCOLOR
: PSET (500 + B, 100 + A), COL: FLAG = 1
: LOCATE 1, 70: PRINT "ON "
END IF
IF INSERT = 0 THEN
: LOCATE 1, 70: PRINT "OFF "
END IF
IF FLAG = 1 THEN PAINT (15 + EB * 12, 15 + EA * 12), COL, WAKUCOLOR
IF EA <> A THEN FLAG = 0
IF EB <> B THEN FLAG = 0
EA = A: EB = B
IF C$ = "0" THEN
PUT (12 + B * 12, 12 + A * 12), X%, PSET
: PAINT (13 + B * 12, 13 + A * 12), COL, WAKUCOLOR
: PSET (500 + B, 100 + A), COL: FLAG = 1
END IF
IF KEY1 = 32 THEN
PUT (12 + B * 12, 12 + A * 12), X%, PSET
: PAINT (13 + B * 12, 13 + A * 12), COL, WAKUCOLOR
: PSET (500 + B, 100 + A), COL: FLAG = 1
END IF
IF KEY1 = 13 THEN
PUT (12 + B * 12, 12 + A * 12), X%, PSET
: PAINT (13 + B * 12, 13 + A * 12), COL, WAKUCOLOR
: PSET (500 + B, 100 + A), COL: FLAG = 1
END IF
IF C$ = "S" THEN GOTO 2000
IF C$ = "s" THEN GOTO 2000
IF C$ = "K" THEN GOTO 5000
IF C$ = "k" THEN GOTO 5000
IF C$ = ";" THEN GOTO 6000
IF C$ = "I" THEN GOTO 7000
IF C$ = "O" THEN GOTO 8000
IF C$ = "P" THEN GOTO 9000
IF C$ = "," THEN GOTO 10000
IF C$ = "." THEN GOTO 11000
IF C$ = "/" THEN GOTO 12000
IF C$ = "-" THEN COL = COL - 1
IF C$ = "*" THEN COL = COL - 1
IF KEY1 = 0 AND KEY2 = 81 THEN COL = COL - 1
IF COL = WAKUCOLOR THEN COL = WAKUCOLOR - 1
IF C$ = "+" THEN COL = COL + 1
IF KEY1 = 0 AND KEY2 = 73 THEN COL = COL + 1
IF COL = WAKUCOLOR THEN COL = WAKUCOLOR + 1
IF COL < 0 THEN COL = MAXCOLOR
IF COL > MAXCOLOR THEN COL = 0
PAINT (620, 40), COL, WAKUCOLOR
LINE (598, 90)-(606, 399), 0, BF
```

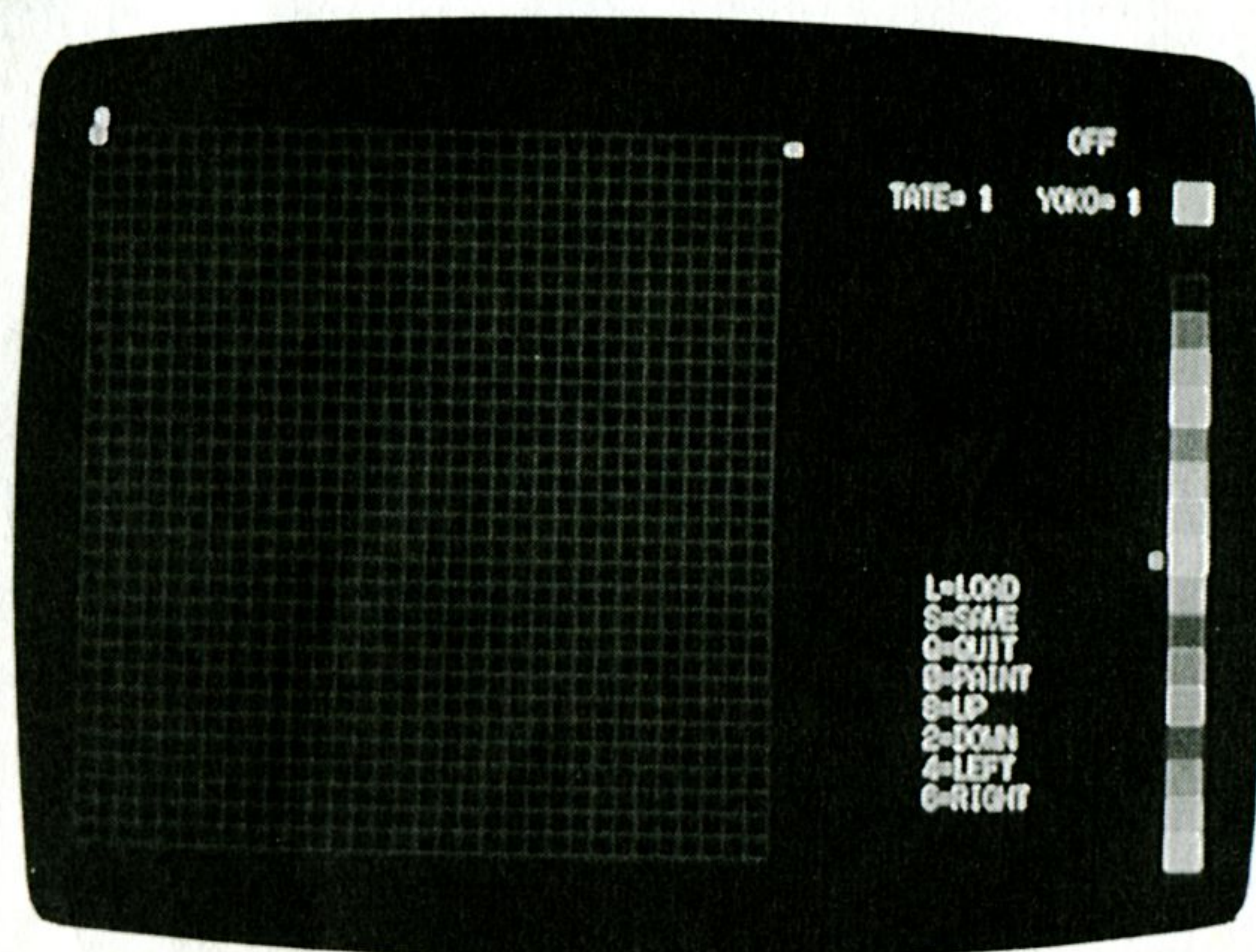
```
 PUT (600, 90 + COL * 20), X%
     ECOL2 = COL
 GOTO 100
 2000
 LOCATE 14, 52: PRINT "                    "
 DIM GRAPH%(300)
 DEF SEG = VARSEG(Z%(0))
 GET (500, 100)-(531, 131), Z%
 OFFSET% = VARPTR(Z%(0))
 LOCATE 13, 52
 PRINT "SAVE"
 2500 LOCATE 14, 52
 INPUT "FILENAME=", NAME$
 IF NAME$ = "" THEN GOTO 2500
 IF NAME$ = "@" THEN END
 LOCATE 2, 52: PRINT "          "
 LOCATE 2, 52: PRINT NAME$
 BSAVE NAME$ + ".GRA", OFFSET%, 800
 LOCATE 13, 52: PRINT "                    "
 LOCATE 14, 52: PRINT "                    "
 GOTO 100
 4000
 LOCATE 13, 52: PRINT "LOAD                "
 LOCATE 14, 52: PRINT "                    "
 PUT (500, 100), F%
 4500 LOCATE 14, 52
 INPUT "FILENAME=", NAME$
 IF NAME$ = "" THEN GOTO 4500
 IF NAME$ = "@" THEN END
 DEF SEG = VARSEG(Z%(0))
 OFFSET% = VARPTR(Z%(0))
 BLOAD NAME$ + ".GRA", OFFSET%
 PUT (500, 100), Z%
 LOCATE 2, 52: PRINT "          "
 LOCATE 2, 52: PRINT NAME$
 FOR I = 0 TO 31
 FOR J = 0 TO 31
 COLOR2 = POINT(500 + I, 100 + J)
  PAINT (15 + I * 12, 15 + J * 12), COLOR2, WAKUCOLOR
 NEXT J
 NEXT I
 LOCATE 13, 52: PRINT "                    "
 LOCATE 14, 52: PRINT "                    "
 GOTO 100
 5000 POSITION$ = "HIDARI  "
      DISPYOKO = 468: DISPTATE = 100
      GOTO DISPLAY
 6000 POSITION$ = "MIGI    "
```

```
     DISPYOKO = 532: DISPTATE = 100
     GOTO DISPLAY
7000 POSITION$ = "HIDARIUE"
     DISPYOKO = 468: DISPTATE = 68
     GOTO DISPLAY
8000 POSITION$ = "   UE   "
     DISPYOKO = 500: DISPTATE = 68
     GOTO DISPLAY
9000 POSITION$ = "MIGIUE  "
     DISPYOKO = 532: DISPTATE = 68
     GOTO DISPLAY
10000 POSITION$ = "HIDARISHITA"
     DISPYOKO = 468: DISPTATE = 132
     GOTO DISPLAY
11000 POSITION$ = "     SHITA"
     DISPYOKO = 500: DISPTATE = 132
     GOTO DISPLAY
12000 POSITION$ = "MIGI SHITA"
     DISPYOKO = 532: DISPTATE = 132
     GOTO DISPLAY
DISPLAY:
LOCATE 13, 52: PRINT POSITION$ + " LOAD "
LOCATE 14, 52: PRINT "                 "
LOCATE 14, 52
INPUT "FILENAME=", NAME$
IF NAME$ = "@" THEN END
DEF SEG = VARSEG(KZ%(0))
OFFSET2% = VARPTR(KZ%(0))
BLOAD NAME$ + ".GRA", OFFSET2%
PUT (DISPYOKO, DISPTATE), KZ%
DEF SEG = VARSEG(Z%(0))
LOCATE 13, 52: PRINT "                    "
LOCATE 14, 52: PRINT "                    "
GOTO 100
RESTART:
LOCATE 15, 54: PRINT "REALY (Y/N)"
LOCATE 15, 65
INPUT ZCHECK$
IF ZCHECK$ = "Y" THEN
A = 0: B = 0
EA = 0: EB = 0
COL = 7
INSERT = 0
GOTO 11110
END IF
LOCATE 15, 54: PRINT "                 "
RETURN
OWARI:
```

```
LOCATE 15, 54: PRINT "REALY (Y/N)"
LOCATE 15, 65
INPUT ZCHECK$
IF ZCHECK$ = "Y" THEN
END
END IF
LOCATE 15, 54: PRINT "                    "
RETURN
```

（プログラムを完全に入力したにもかかわらず動作しない場合は，11ページの囲み，および90ページのコラムを参考にしてみてください）

★キャラクタジェネレータの使い方

キャラクタジェネレータの使い方は簡単です．まず，このプログラムを正しく入力，保存した後実行してみてください．次のような画面が現れたはずです．

★カラー口絵
p.1参照

どんなキャラクタでも作れるキャラクタジェネレータ

このプログラムを実行すると，32×32のマス目が現れます．このマスのうち，一番左上に小さな四角が見えます．この四角がカーソルの位置です．カーソルの位置は右上にも数字で表示されています．このカーソルは，数字キーにより次のように動きます．

キャラクタジュネレータのカーソルとカラー

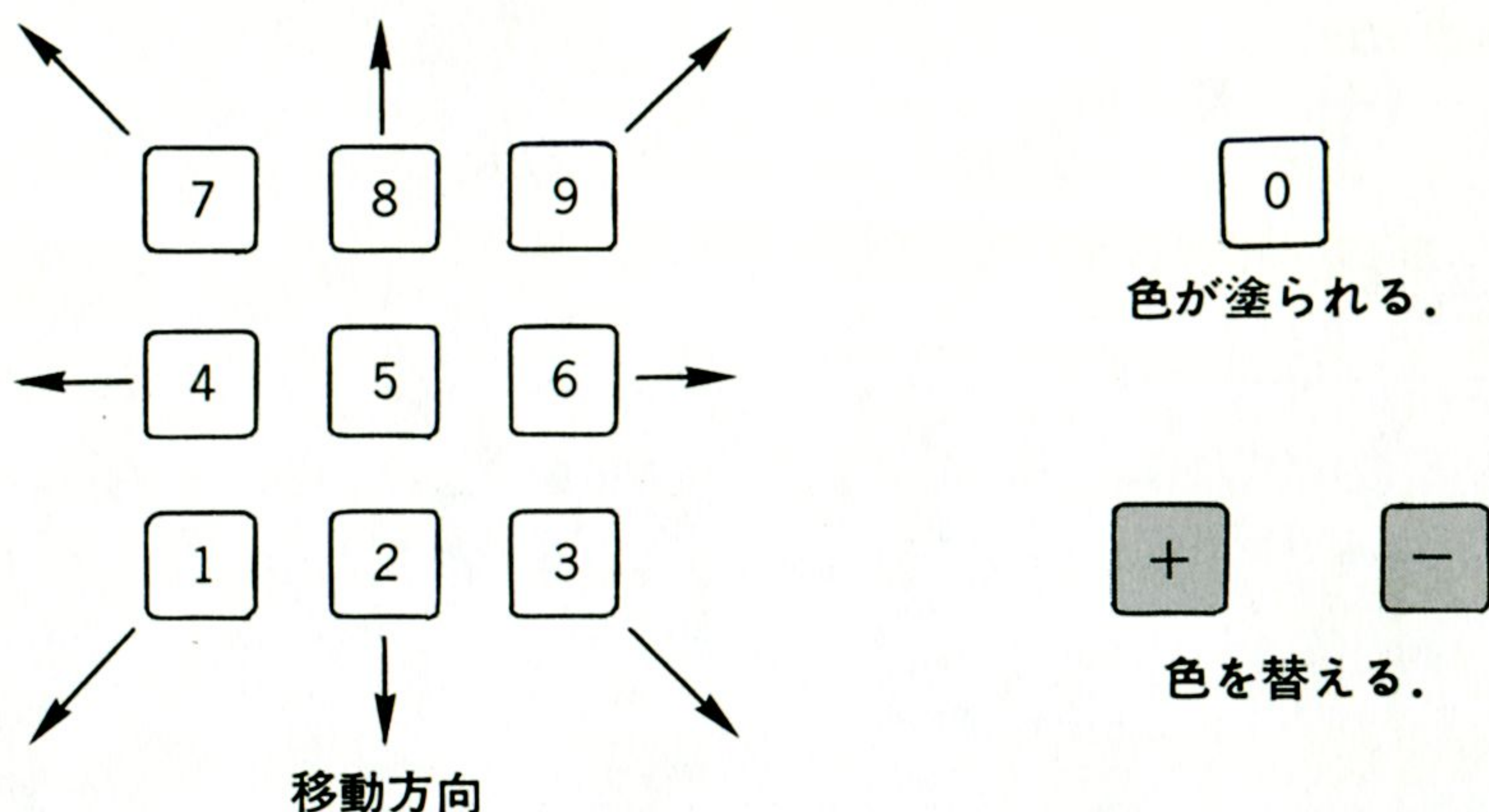
7	8	9
4	5	6
1	2	3

移動方向

0

色が塗られる.

＋　－

色を替える.

右側に縦に四角が並んで，さまざまな色を示しています．この色のうち，左に小さな四角が現れている色，または一番上の四角の中の色がそのときの色です．

この色は，『＋』または『－』またはROLL UP，ROLL DOWNを押すことにより，色が並んでいる順番に1つずつ変わっていきます．ただし，1色だけは線をかくのに使っているために使えません．カーソルのある位置で数字の『0』またはリターンキー，スペースキーを押すと，カーソルのあるマス目が，その時に選ばれている色で塗られます．

『INS』(インサート)キーを押すと，画面右上部にONが表示され，連続ペイントモードがONになります．

このモードですと，いちいち『0』を押さなくてもカーソルが移動したところが，選ばれた色で塗られて行きます．

『DEL』(デリート)キーを押すと，画面上部にOFFが表示され，元のモードにもどります．

これを使って，32×32のマス目の中に，自分の好きな形を自分の好きな色でかいてください．

絵をかき終えたり，途中で一端中止したくなったときは『S』を押してください．すると画面の右中央に

SAVE_FILENAME_=

の文字が現れます，8文字までのアルファベットで始まる英数字を入れて

ENTER キーを押してください．絵は，入力した名前 .GRA のファイル名でSAVE されます．この名前は，あとで絵を呼び出すときに必要ですから，しっかり記録しておいてください．

かきかけの絵を完成させたり，いままでかいた絵を修正して別の絵にしたいときは，『L』を押してください．すると，画面の右中央に LOAD_FILENAME_＝が現れます．ここで自分が呼び出したいファイルの名前を入力した後，ENTER キーを押してください．自分の呼び出したい絵が現れるはずです．なおこのときファイル名に．GRA は付ける必要はありません．

Bドライブからセーブやロードをしたいときには，ファイル名を聞いてきたら B： を名前の前に付けてあげてください．

『K』を押すと，FILE NAME を聞いてきます．ファイルの名前を入力すると，Lで呼んだ絵の左横にKで呼んだ絵が表示されます．

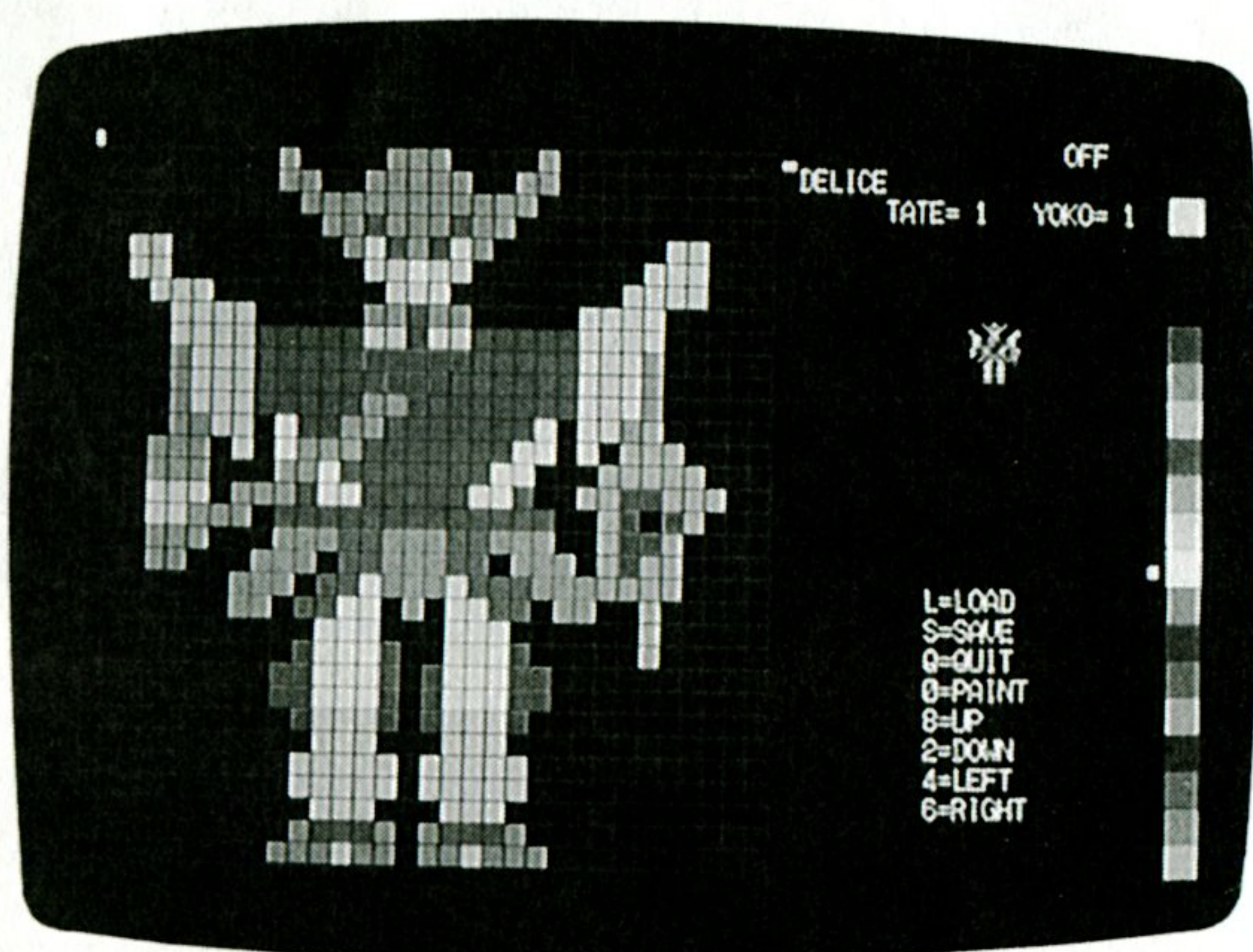
同じように，上下も含めて『L』キーの周りのキーを押すと，そのキーに対応した場所に絵をロードすることができます．これは，城などの連続した絵をかくときに，隣りがどうなっているのかを見るのに便利です．

『Z』を押すと，『REALY(Y/N)』と聞いてきます．『Y』と答えると，すべての絵を消して，また最初からかき始めることができます．

『Q』で終了します．

このキャラクタジェネレータを使えば，自分の作りたいキャラクタが何でもできます．

思いっきり楽しい絵をたくさんかいてください．

OFF
DELICE
TATE= 1 YOKO= 1
L=LOAD
S=SAVE
Q=QUIT
0=PAINT
8=UP
2=DOWN
4=LEFT
6=RIGHT

★カラー口絵
p.1参照

キャラクタジェネレータでロボットを作る

Quick BASIC

1億円は金庫の中，机の上？

もし，1億円を現金で持っていたら，なくなりはしないか？　と心配で心配で，しょうがないでしょう．当然，金庫にしまっておくはずです．中には腹巻に巻いてお腹にかかえているという人もいるとは思いますが(でも，ゴツゴツして寝られないのじゃないかな？)．

せっかく作った無敵のキャラクタ金庫に入れて保管しよう
(英語でシャレルと，セーフにセーブ！　なんちゃって)．

せっかくかいたキャラクタですから，きちんとディスクにしまっておきましょう．世の中，自分の作ったデータほど貴重なものはありません．
かくいう筆者も，昔，パソコンかけ出しのころ，金欠病のあまり高いソフトウェア本体のみをバックアップし，データディスクをバックアップしておかず，苦労したことがあります．
なんせ，フロッピーディスク1箱7000円以上の時代だったのです．
考えて見れば，市販のプログラムソフトはお金を出せばまた買えますが，自分の作ったデータは，なくなってしまえばいくらお金をつんでももどりません．また，同じ労力が必要となるだけです．御用心，御用心．
キャラクタの保存(セーブ)は，キャラクタジェネレータを使えば簡単です．
セーブしたいときに『S』を押してください．
セーブするファイル名を聞いてくるので，英字で始まる8文字以内の英数字を入れてください．リターンキーを押すと自動的に拡張子.GRAが付いた名前でセーブされます．

このとき，くれぐれも名前をメモしておいてください．また，前にセーブしたファイルと同じ名前を入れると，前のファイルを消してしまうので注意してください．

ここでは，キャラクタのセーブ(保存)の原理について学んで見ましょう．
100ページのCGENE. BASの17行目を見てください．
まず，

```
DIM_Z%(3000)
```

で，メモリ上に絵を取り込むべき場所を配列として確保しています．Zのあとの%は，整数型の変数として確保することを示しています．(　　)の中の数字3000により，整数型3000分のメモリを確保するように命令しています．
次に，103ページ8行目の

```
GET_(500,_100)−(531,_131),_Z%
```

で，画面の左上が（500，100）の点，右下が（531，131）の点で囲まれる四角形の中の絵を，先程メモリ上に確保したZ%の部分のメモリに入れています．

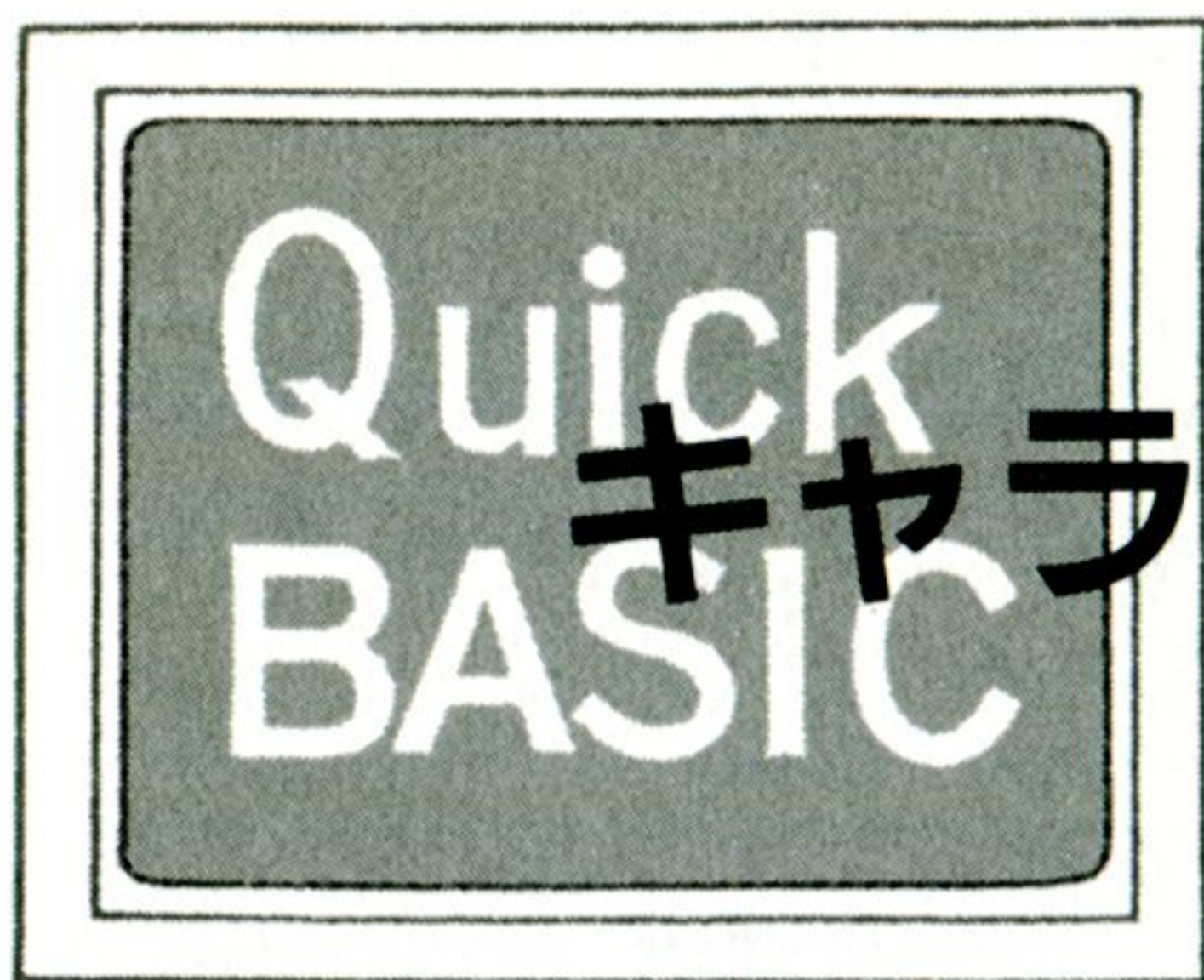
```
DEF_SEG_=_VARSEG(Z%(0))
```

の行で，メモリのセグメントをZ%のメモリの一番最初の値にセットしています．

```
OFFSET%_=_VARPTR(Z%(0))
```

で，参照すべきメモリの最初を，先程のZ%のメモリの1番最初の値にセットしています．セグメント，オフセットという2つの値が出てきましたが，この2つを使って，メモリ上の正確な位置を指定しています．

```
BSAVE_"NAME$_+_". GRA",_OFFSET%,_800
```

で，先程選んでおいたメモリの場所，つまり，絵を収納したZ%の先頭から800バイト分をNAME$. GRAという名前で，ファイルとしてディスク上にセーブしています．NAME$は，そのとき自分が入力したアルファベットで始まる8文字以下の名前で，PRINCESSでもMOBILSTでも構いません．これで，フロッピーディスクの上に，先程の絵がPRINCESS. GRAのようにセーブされます．
これらの話を倉庫会社の話にたとえると，まず，DIM命令で荷物を入れる箱を作ります．次に，GET命令で絵をいま準備した箱に入れます．DEFSEGとVARPTRが，倉庫会社に教える荷物を取りに来る住所を示しています．
次に，BSAVE以下の命令で，取りに来た荷物をフロッピーディスクという倉庫にPRINCESS. GRAという名札を付けてしまっています．

Quick BASIC キャラクタを動かす！

どうですか？　自分の好きなキャラクタができましたか．いつまでも箱だけ動かしていてもつまりません．今度は，作ったキャラクタを動かしてみましょう．

★カニの親子

カニのお母さんがいいました．
「ぼうやみっともないからそんな横に歩いちゃいけないよ」
カニのぼうやがいいました．
「お母さんお手本をみせてちょうだいな」

YOKOA
=YOKOA+1

YOKOB
=YOKOB+1

カニの親子

次の IDOU.BAS のプログラムを入力，保存してください．そのとき，5行目の　NAME$_=_"HERO"　の"（ダブルクォーテーションマーク）で囲まれた中の名前は，自分で作って SAVE したキャラクタの名前を入れてください．

IDOU．BASプログラム

```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
TATE = 200: YOKO = 320
NAME$ = "HERO"
DIM HERO%(800)
    DEF SEG = VARSEG(HERO%(0)): OFFSET% = VARPTR(HERO%(0))
    BLOAD NAME$ + ".GRA", OFFSET%
    DEF SEG
PUT (YOKO, TATE), HERO%
DO
BEGIN:
A$ = INKEY$
IF A$ = "" THEN GOTO BEGIN
    IF A$ = "1" THEN TATE = TATE + 32: YOKO = YOKO - 32
    IF A$ = "2" THEN TATE = TATE + 32
    IF A$ = "3" THEN TATE = TATE + 32: YOKO = YOKO + 32
    IF A$ = "4" THEN YOKO = YOKO - 32
    IF A$ = "6" THEN YOKO = YOKO + 32
    IF A$ = "7" THEN TATE = TATE - 32: YOKO = YOKO - 32
    IF A$ = "8" THEN TATE = TATE - 32
    IF A$ = "9" THEN TATE = TATE - 32: YOKO = YOKO + 32
    IF A$ = "Q" THEN END
    PUT (YOKO, TATE), HERO%, PSET
LOOP
```

原理は，前に出て来た箱の移動であるHAKO.BASとあまり変わりません．違うところは5行目から10行目までで，前にCGENE.BASを使って作ってSAVEしておいた絵をメモリに読み込み，それをYOKO,TATEという位置にPUTで表示しているところです．同じように，24行目でLINE文で箱をかく代わりにPUT命令で先程読み込んだ絵を表示させています．PUT命令の最後のPSETは，別名“ヤッチャン命令”といいます．これは，いままでそこに誰がいようとどけて自分の絵をかく命令です．このほかにも，“仲良し命令”，“ご遠慮命令”と各種そろっています．

では，このプログラムを実行して見ましょう．

実行すると，画面の中央に自分が作ったキャラクタが現れたはずです．数字キーの1～9（5を除く）を押すと，押した方向に動くはずです．

アレッアレッ，でも何か変ですね．

画面の中でキャラクタがどんどん増えてしまいました．移動するたびに新しい位置にキャラクタが出ますが，前のキャラクタが残ってしまいます．忍術の世界での分身の術では歓迎されますが，ゲームの世界での分身の術は歓迎されません．ゲームの世界では“困った君(くん)”となってしまいます．

そこで，次の IDOU2.BAS のように改良してください．

IDOU2．BASプログラム

```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
DIM SHOUKYO%(800)
GET (0, 0)-(31, 31), SHOUKYO%
YOKO = 320: TATE = 200
OLDYOKO = YOKO: OLDTATE = TATE
NAME$ = "HERO"
DIM HERO%(800)
    DEF SEG = VARSEG(HERO%(0)): OFFSET% = VARPTR(HERO%(0))
    BLOAD NAME$ + ".GRA", OFFSET%
    DEF SEG
PUT (YOKO, TATE), HERO%
DO
BEGIN:
A$ = INKEY$
IF A$ = "" THEN GOTO BEGIN
    IF A$ = "1" THEN TATE = TATE + 32: YOKO = YOKO - 32
    IF A$ = "2" THEN TATE = TATE + 32
    IF A$ = "3" THEN TATE = TATE + 32: YOKO = YOKO + 32
    IF A$ = "4" THEN YOKO = YOKO - 32
    IF A$ = "6" THEN YOKO = YOKO + 32
    IF A$ = "7" THEN TATE = TATE - 32: YOKO = YOKO - 32
    IF A$ = "8" THEN TATE = TATE - 32
    IF A$ = "9" THEN TATE = TATE - 32: YOKO = YOKO + 32
    IF A$ = "Q" THEN END
    PUT (OLDYOKO, OLDTATE), SHOUKYO%, PSET
    PUT (YOKO, TATE), HERO%, PSET
    OLDYOKO = YOKO: OLDTATE = TATE
LOOP
```

まず変わったところは，4行目と5行目の

```
DIM_SHOUKYO%(800)
GET(0,_0) − (31,_31),_SHOUKYO%
```

です．これは，SHOUKYO%という配列を宣言して，そこに何もかいていない絵の部分を取り込んでいます．

次に7行目の

OLDYOKO_=_YOKO:_OLDTATE_=_TATE

で絵を消すときに，どこを消すかという値を最初に表示する値にそろえています．

27行目で新しく絵をかく直前に，いままであった絵のところになにもない絵SHOUKYO%を表示して，いままでの絵を消しています．

絵の消去は，新しい絵をかく直前でやった方が絵のチラつきが少なくてすみます．

次の行で新しい絵を表示した後，29行目で，

OLDYOKO_=_YOKO:OLDTATE_=_TATE

として，いま絵をかいた場所を，それぞれ OLDYOKO, OLDTATE という変数に入れて，次に消すときにはどこを消せば良いか教えています．

では，このプログラムを実行してみてください．うまくいきましたか，では拍手….

エッ，いらない絵は消えるけど，はじまで行くと，

引数が許される範囲ではありません (ERR=5)

のエラーが起こって止まってしまうですって．

そうなのです．原子炉の周り，基地の周り，ジャイアンツが負けた翌日の某君の周りと，世の中，立入禁止の場所がいっぱいあります．実はプログラムの世界にも立入禁止の場所があったのです．

前の HAKO.BAS のときのように LINE 命令で絵をかいている場合には，横0～639，縦0～399の点の範囲を外れて絵をかいてもなにも起こりませんでした．ところが，PUT 命令で絵を表示する場合，絵の一部分でも上記をはみ出る範囲に絵を表示しようとすると，このエラーが発生してしまいます．

これを防ぐには，次のような命令を，先程の IDOU2.BAS プログラムの26行目　IF_A$_=_"Q"_THEN_END　のあとに付け加えてください．

```
IDOU3．BASプログラム
  IF TATE < 0 THEN TATE = 0
  IF TATE > 367 THEN TATE = 367
  IF YOKO < 0 THEN YOKO = 0
  IF YOKO > 608 THEN YOKO = 608
```

これでキャラクタは安心して画面の中を移動できます．自分の作ったキャラクタが動き回るってすごく楽しいと思いませんか？

では，次に先程のプログラムの数字を，次のように替えて保存，実行して見てください．

```
IDOU4．BASプログラム
  IF TATE < 0 THEN TATE = 367
  IF TATE > 367 THEN TATE = 0
  IF YOKO < 0 THEN YOKO = 608
  IF YOKO > 608 THEN YOKO = 0
```

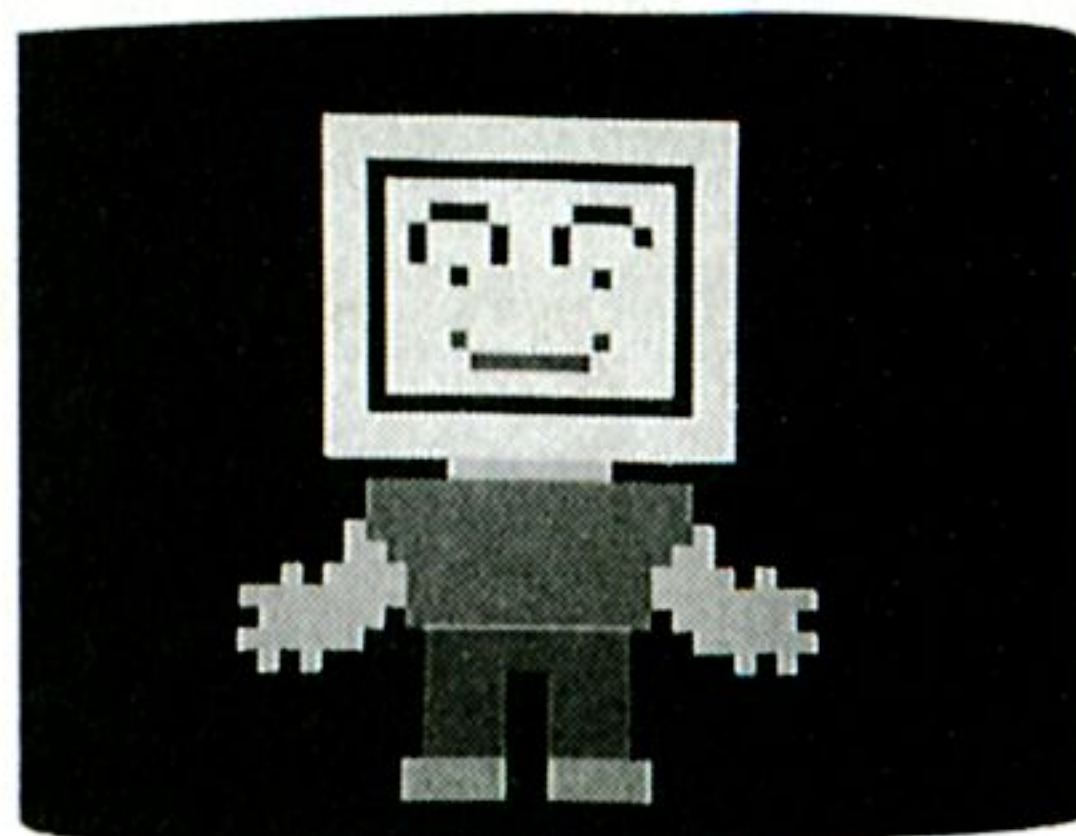
今度は，画面の一番下までキャラクタが来ると一番上から，左に来ると右から出てきます．

シューテングゲームなどに使っても，おもしろいものができるかも知れません．いろいろなキャラクタで楽しんでください．

(プログラムを完全に入力したにもかかわらず動作しない場合は，11ページの囲み，および90ページのコラムを参考にしてみてください)

★カラー口絵 p. 2参照

「My name is HERO.」

Quick BASIC

保存はまとめて面倒みます

★画面をまとめて保管しよう

画面全部を一度にセーブ／ロードする方法について述べてみたいと思います．

パソコンの絵は，画面上の赤，青，緑の多数の点のうち，どの点をどれぐらいの強さで光らせるかによって，絵をかいています．そこで，赤を表示するためのメモリ上にある数値を順序よくディスク上のファイルに保存しておき，必要に応じてこのディスク上のファイルを，赤を表示するためのメモリにもどしてやれば，赤の面を表示することができます．

これと同じ操作を，青と緑についても行えば，画面上の絵の保存と呼出しができるわけです．

次のプログラム GAMENSVE. BAS を入力，実行してみてください．

```
GEMENSVE. BASプログラム
 SCREEN 88, 0, 1, 1
 WINDOW SCREEN (0, 0)-(639, 399)
 CLS
 CIRCLE (200, 200), 70, 1
 PAINT (200, 200), 1, 1
 CIRCLE (300, 200), 70, 2
 PAINT (300, 200), 2, 2
 CIRCLE (400, 200), 70, 4
 PAINT (400, 200), 4, 4
 NAME$ = "SAMPLE"
 FOR I = 1 TO 3
 IF I = 1 THEN SAVENAME$ = NAME$ + ".B": VRAM = &HA800
 IF I = 2 THEN SAVENAME$ = NAME$ + ".G": VRAM = &HB000
 IF I = 3 THEN SAVENAME$ = NAME$ + ".R": VRAM = &HB800
 DEF SEG = VRAM
 BSAVE SAVENAME$, &H0, &H7D00
 NEXT I
```

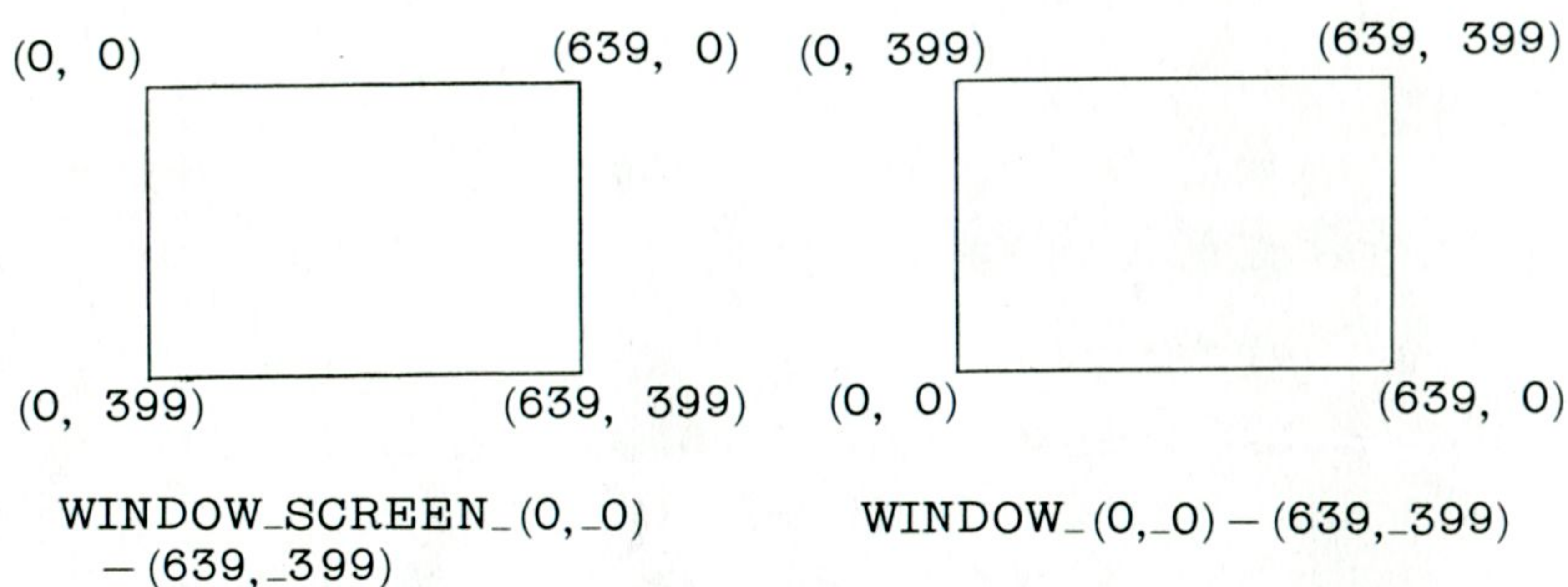
1行目，SCREEN_88…は，画面を8色モードで使うことを示しています．

2行目は，画面の左上の点を0，0右下の点が639，399になるように使うということを示しています．SCREENと書かずにWINDOWのみだと，左下の点が0，0右上の点が639，399となります．

(0, 0)　(639, 0)
(0, 399)　(639, 399)

WINDOW_SCREEN_(0,_0)
–(639,_399)

(0, 399)　(639, 399)
(0, 0)　(639, 0)

WINDOW_(0,_0)–(639,_399)

3行目，CLSは，それまで画面にある不要な字や絵を消しています．

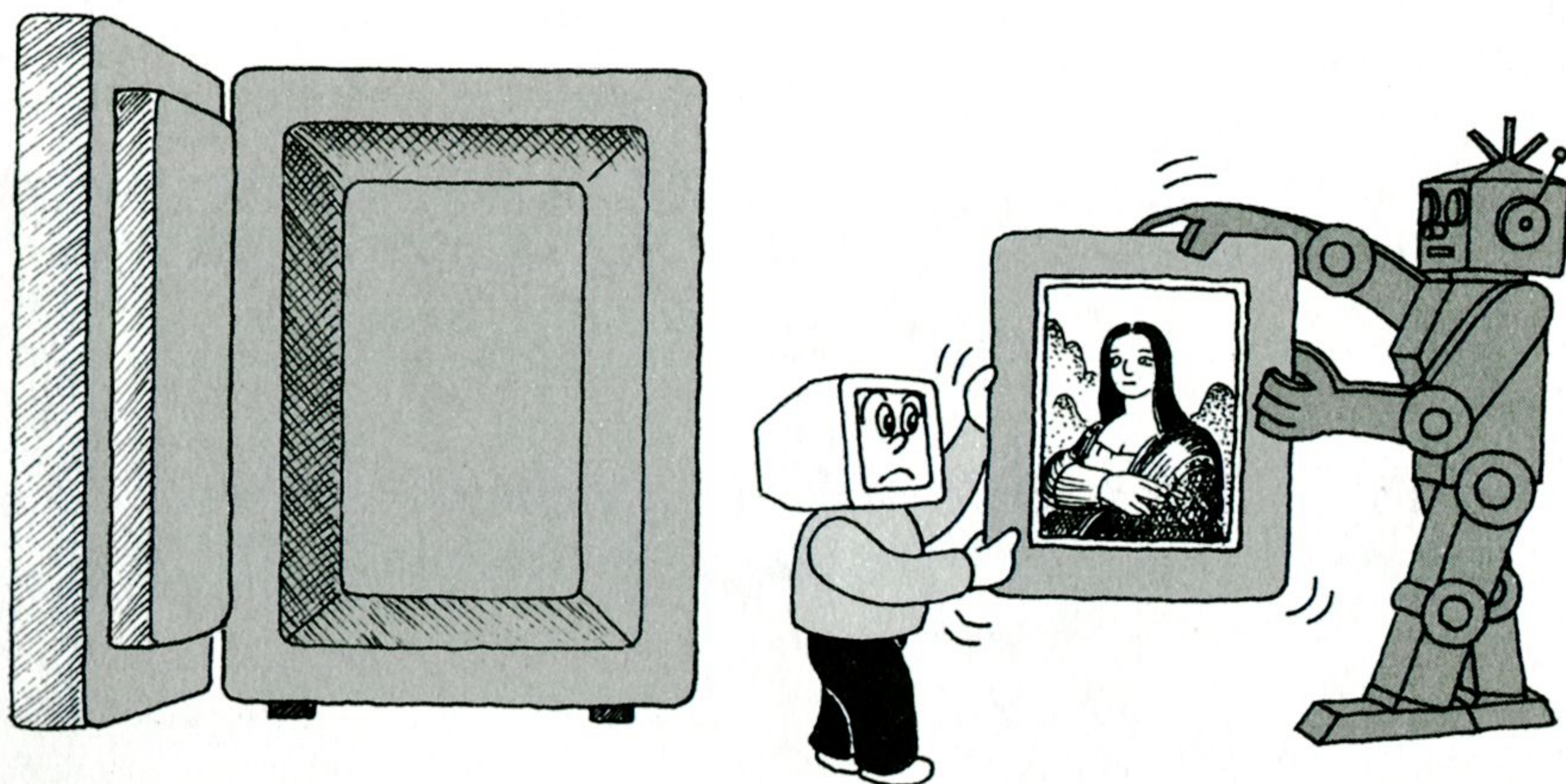
4行目から9行目までは，何の絵もないと，きちんと保存・呼出しができたかわからないので，単に円を描いて，その中を色で塗っているだけです．

10行目で保存するための名前を決めています．ここでは単にSAMPLEとしていますが，新しい絵を保存するたびに適当に名前を変えてください．同じ名前を使うと，前に保存してあった絵を消してしまいます．

12行目からの3行で，それぞれの色に対応してB，G，Rを付けています．つまり，青の画面はSAMPLE. Bという名前のファイルに，緑の画面をSAMPLE.G，赤の画面をSAMPLE.Rというファイルにしています．また，後ろにある&HA800などは，それぞれの色の画面の始まりのメモリのセグメントを示しています．

16行目のBSAVE命令で，実際に画面を保存しています．

このプログラムを他の場面で応用するためには，まず絵をかく前に，プログラムの先頭にこのプログラムの3行目までを書いておきます．次にLINE文などで自分のほしい絵をかいたあと，プログラムの最後に10行目以後の文を書きます．ただし，10行目の”　　”で囲まれた中の名前は，絵ごとに必ず変えてください．また，この名前は，呼び出すときに必要ですから，どこかに記録しておいてください．

このプログラムは8色モードで作動します．

★画面の呼出し

せっかく画面を保存してもこれを呼び出せないのでは，まるっきり金庫に現金を入れてカギをなくしたようなものです．これでは困ってしまいます．そこで，画面を呼び出すプログラムを作りましょう．Quick BASIC 上で新規を選んだあと，次のプログラム GAMENLDE. BAS を入力してください．

GEMENLDE．BASプログラム

```
SCREEN 88, 0, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
NAME$ = "SAMPLE"
FOR I = 1 TO 3
IF I = 1 THEN LOADNAME$ = NAME$ + ".B": VRAM = &HA800
IF I = 2 THEN LOADNAME$ = NAME$ + ".G": VRAM = &HB000
IF I = 3 THEN LOADNAME$ = NAME$ + ".R": VRAM = &HB800
DEF SEG = VRAM
BLOAD LOADNAME$
NEXT I
```

最初の3行までは，前の節で説明したことと同じです．4行目で呼び出す画面の名前を決めています．”　　”で囲まれた中の名前を，自分が保存した画面に付けたさまざまな名前にすることで，いろいろな画面を呼び出すことができます．

6行目から8行目までで，各色の画面の名前と，そのファイルの中のデータをメモリのどの部分に送れば良いかを決めています．

9行目でファイルの取込み先を，その直前で決めたメモリのセグメントにセットしています．

10行目のBLOAD命令がファイルをメモリに取り込むための命令です．

このプログラムを応用する場合，他のプログラムの中で実際に画面を呼び出したい部分に，このプログラムと同じことを書いておきます．ただし，4行目の”　　”に囲まれた名前だけは，前に保存するときに記録しておいた名前にします．

このように，画面全体を保存，呼び出す方法は，最初からその場で複雑な絵をかくのよりも絵をかく時間が短くて済む利点があります．また，1番の利点としては，たった数行で複雑な絵を表示できる点にあります．

ところで，さっきのプログラムでは，絵の表示を各色ごとに上からやってい

コラム

特別大付録，IBM PCでの画面の扱い方

DOS Vも5.0版が出荷され，日本でもIBM PCが本格的に活躍しようとしています．

私もIBM PCを1台持っていますが，アメリカの最新のゲームを人より早く使えたり，また，日本よりソフトウェアがずーと安く買えたりと，結構楽しんでいます．これからもIBM PCがどんどん増えて行くと思います．そこで，IBM PC上でのQuick BASICの使い方をここで述べておこうと思います．

VGA対応の場合，SCREENモードは12です．また，ドット数が640×480なので，画面を扱うプログラムの最初の3行は次のようにします．

```
SCREEN_12
WINDOW_SCREEN_(0,_0) - (639,_479)
CLS
```

画面を保存するには，次のIBMGSAVE.BASを，絵をかいたプログラムの最後に付け加えてください．また，”　　”で囲まれたFILE名は適当に替えてください．

ました．でも，これではあまりカッコ良くありません，そこで，次のように修正してください．

GEMENLD2．BASプログラム

```
SCREEN 88, 0, 1, 0
WINDOW SCREEN (0, 0)-(639, 399)
CLS
NAME$ = "SAMPLE"
FOR I = 1 TO 3
IF I = 1 THEN LOADNAME$ = NAME$ + ".B": VRAM = &HA800
IF I = 2 THEN LOADNAME$ = NAME$ + ".G": VRAM = &HB000
IF I = 3 THEN LOADNAME$ = NAME$ + ".R": VRAM = &HB800
DEF SEG = VRAM
BLOAD LOADNAME$
NEXT I
SCREEN 88, 0, 0, 1
```

IBMGSAVE．BASプログラム

```
SCREEN 12
WINDOW SCREEN (0, 0)-(639, 479)
bytes& = 38400
FOR I% = 0 TO 3
     OUT &H3CE, 4
     OUT &H3CF, I%
     BSAVE "AAA" + ".BP" + CHR$(49 + I%), 0, bytes&
NEXT I%
```

画面をロードしたい場合，次の IBMGLD.BAS をロードしたい部分に書いてください．ただし，ファイル名はセーブしたときと同じものを使ってください．

IBMGLD．BASプログラム

```
FOR I% = 0 TO 3
     OUT &H3C4, 2
     OUT &H3C5, 2 ^ I%
     BLOAD "AAA" + ".BP" + CHR$(49 + I%)
NEXT I
```

IBM PC で動かすには，IBM PC 用の Quick BASIC が必要です．

上記のやり方は，IBM 版の DOS 5.0 および IBM PC 用の Quick BASIC で使っています．また，VGA 以外のモードには対応していません．

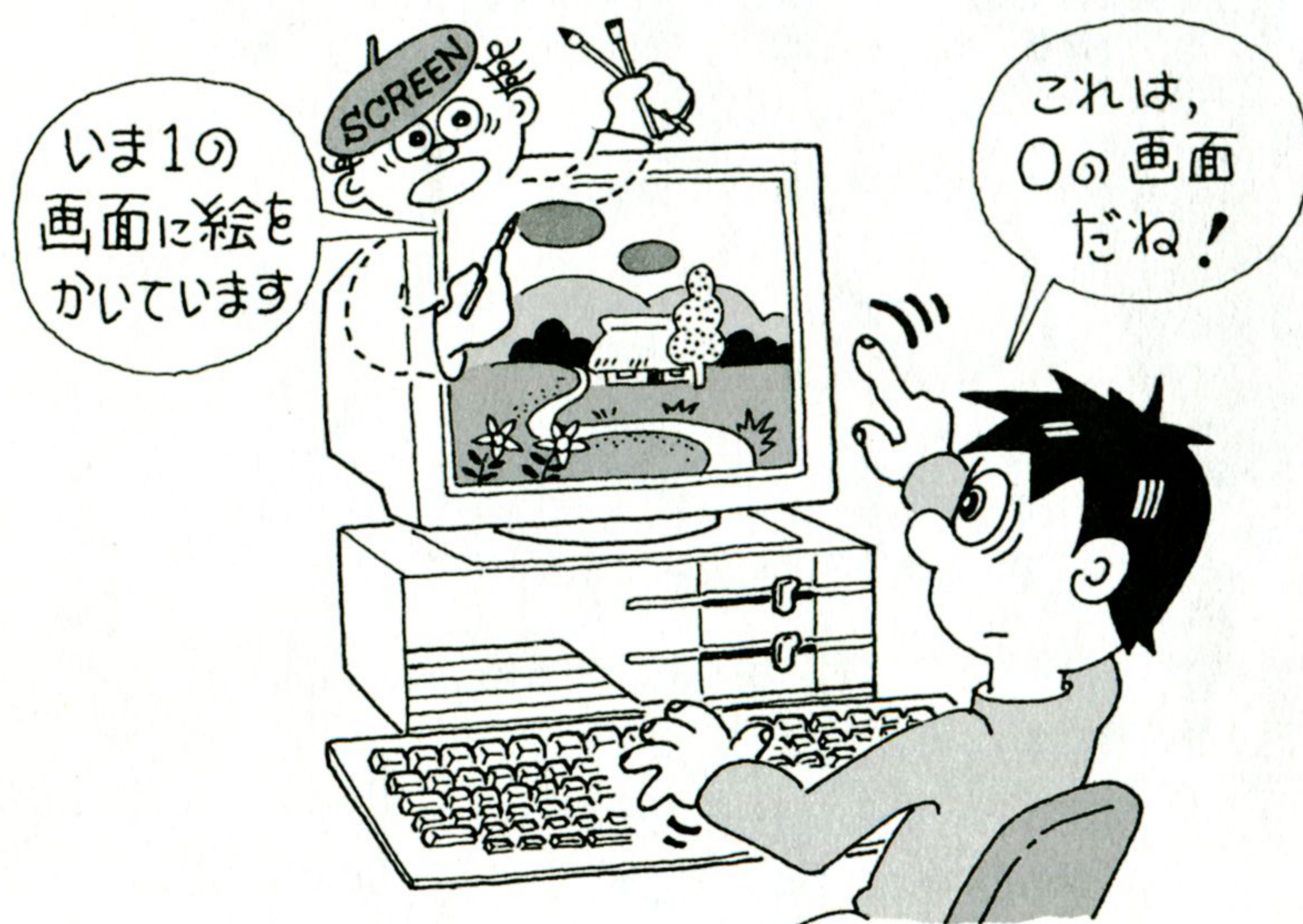
変わったところは，1行目と最後に1行付け足したところです．いまのPC-9801は，0と1と画面を2画面持っています．

ここで，いまのプログラムの1行目のSCREEN文ですか，最後から2つ目の1で，絵をかくのを1の画面で行っています．また，一番最後の0で，そのとき実際に見えているのは0の画面です．こうやって見えないところで絵をかいておき，かき終わったところで，最後の行で1の画面を表示しています．市販のほとんどのゲームが，絵を全面的にかき替えるときには，この方法を使っています．

モード　色数　絵をかくアクティブ画面　表示をする画面

```
SCREEN 88, 0, 1, 0
```

いままでに見てきた画面セーブのプログラム部分を，絵をかくプログラムのうしろに書いて実行すると，自分の絵がディスクにセーブされます．また，画面ロードのプログラム部分を絵を表したいプログラムに入れておくと，ディスクに入れておいた絵をいつでも読み出すことができます．

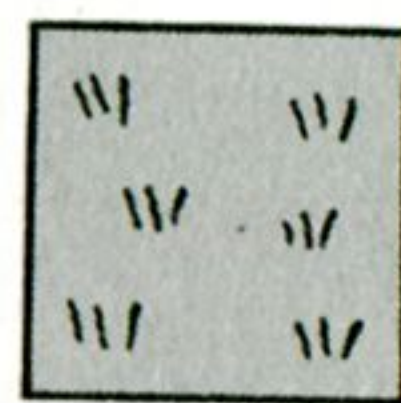
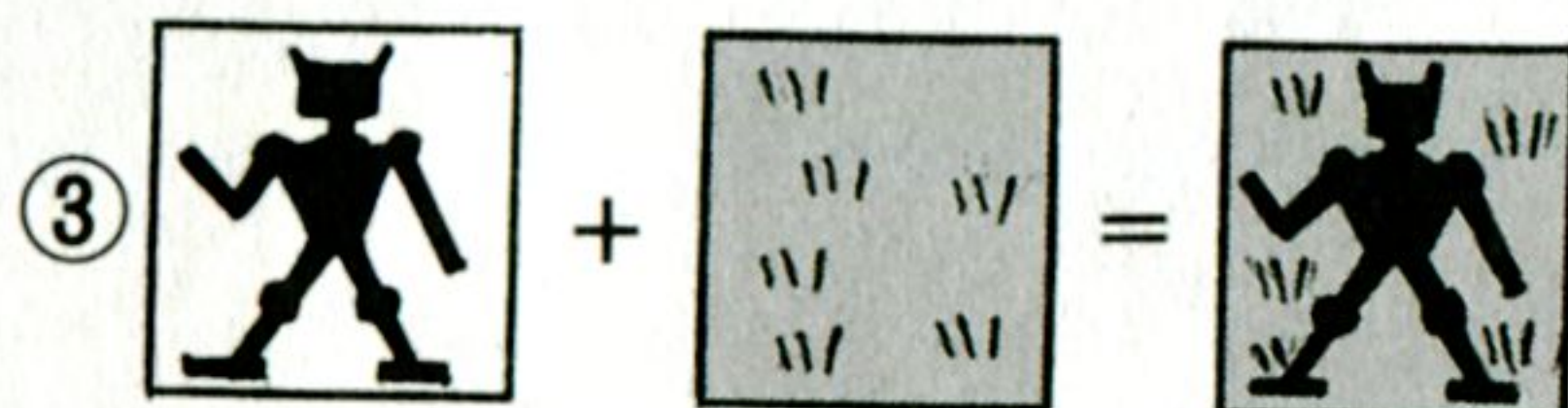
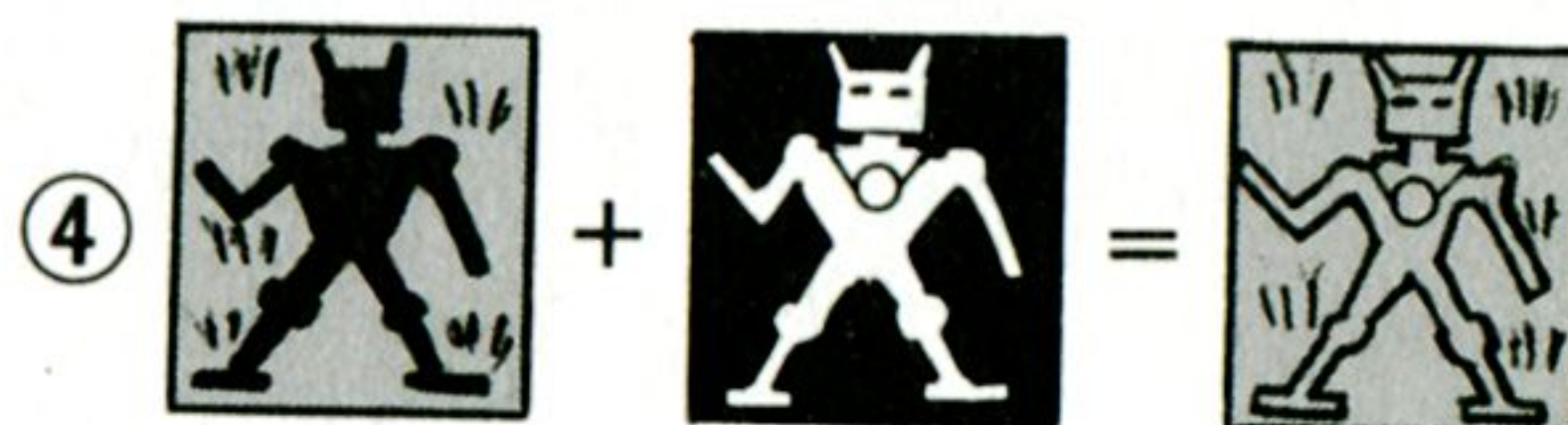
このプログラムは8色モードで作動します．

Quick BASIC “重ね合わせ”は難しい?

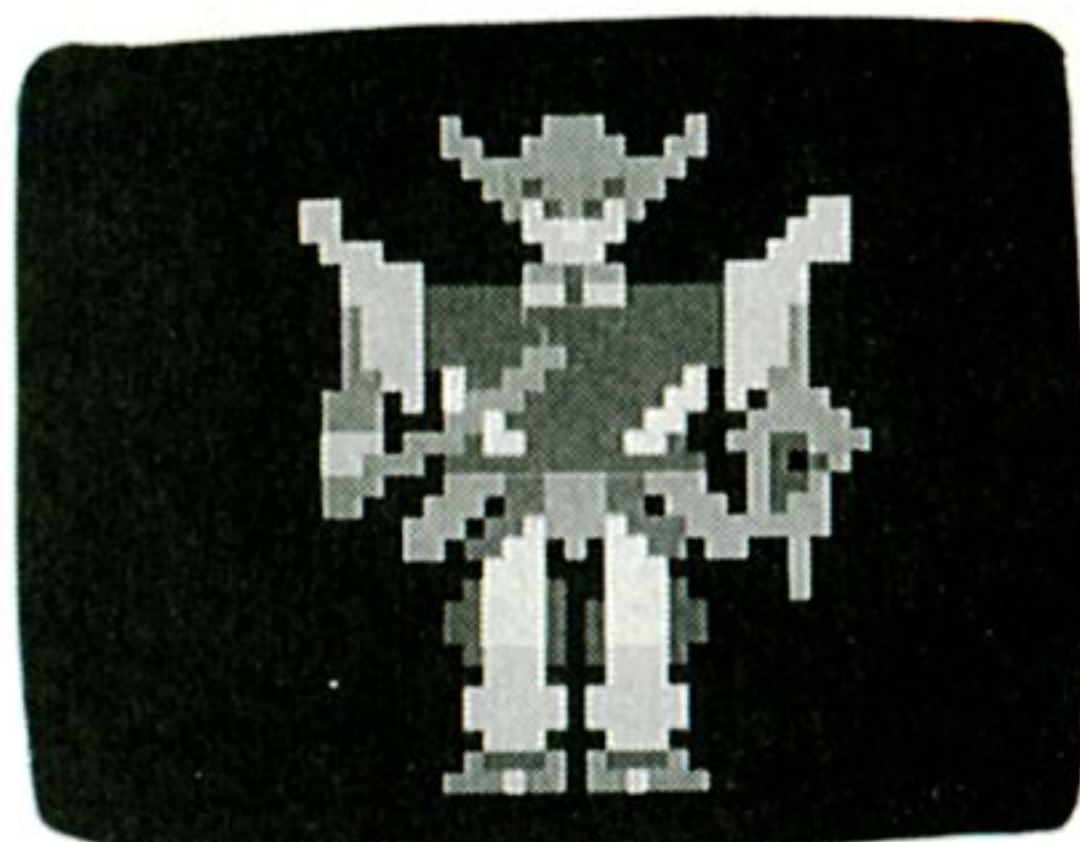
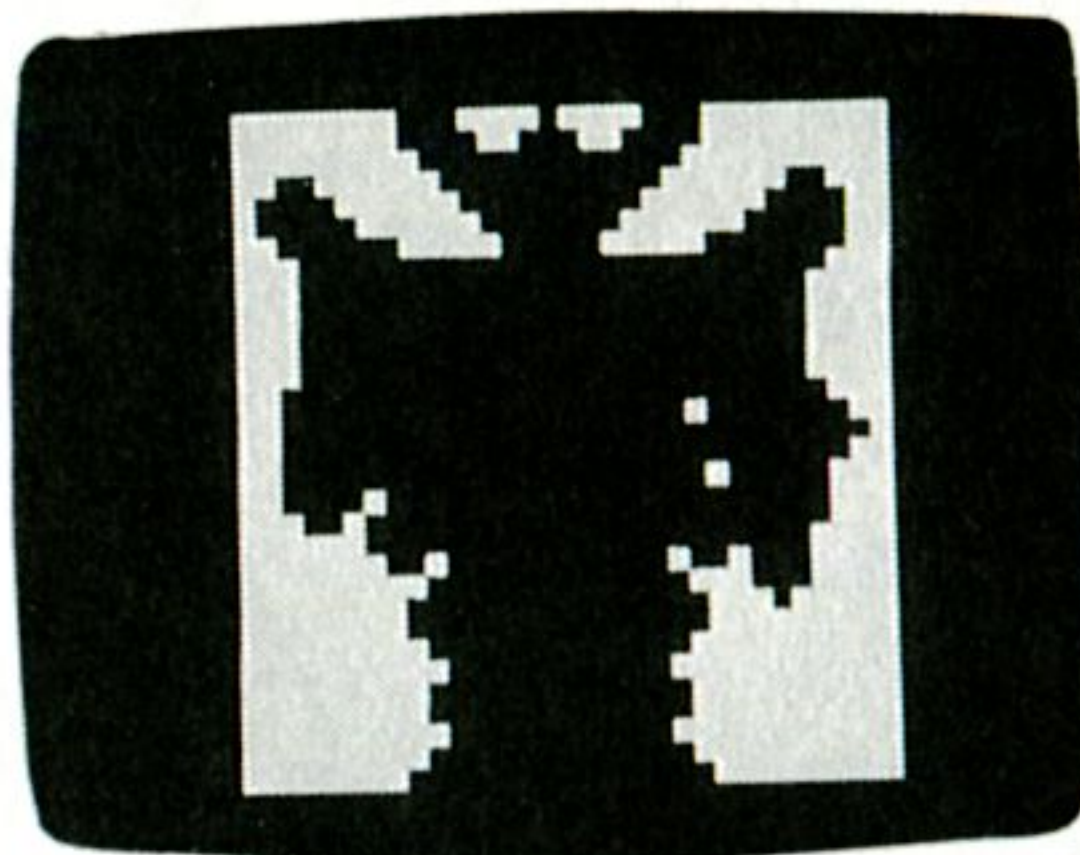
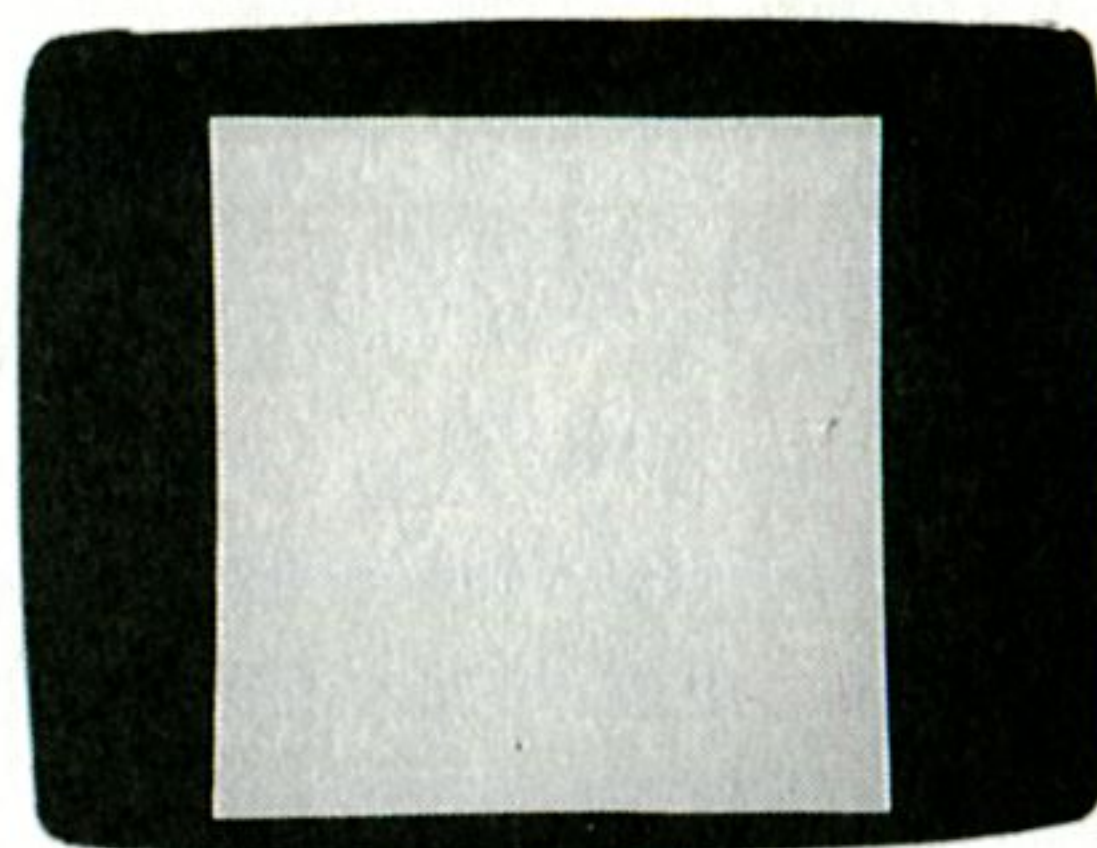
“重ね合わせ”というのは，背景や敵のキャラクタの上に味方のキャラクタを表示したとき，両方の絵が混ざり合わず，自分の表示したい絵が優先され，その向う側に背景などが見えるようにするテクニックです．

ファミコンなどのゲーム専用機や，一部のパソコンでは，最初から“重ね合わせ”を考えてハードウェアが作ってあり，背景面に絵をかいておき，手前のスプライト面にキャラクタを描けば，パソコンがかってに“重ね合わせ”をやってくれます．それに引き替え，PC-9801やIBM PCでは，この機能がなく，プログラムを作る私達が面倒を見てやらなくてはいけません．

ここでは，“重ね合わせ”の技術について学びましょう．

★本格的重ね合わせ

次の本格的重ね合わせの図を見ながら読んでください．

本格的重ね合わせ

★カラー口絵p.1参照

主人公DELICE(デライス)　TOUMEI(透明)　PINK(全面ピンク)
GREEN(全面録)

まず，キャラクタの絵DELICE. GRAを用意します．CGENEを使えば簡単です．作り終わって絵を保存したところで，透明の絵を作ります．つまり，キャラクタの絵のあるところは真っ黒で，それ以外のところは真っ白の絵です．この絵と背景を重ね合わせると，白いところだけ背景が透けて見えるので，透明と呼ぶことにします(黒いところに着目すれば,影またはマスクということになります)．CGENE.BASプログラムで作って，TOUMEIという名称で保存しておいてください．

次に，全部をピンクに塗ったPINK. GRAと緑に塗ったGREEN. GRAの背景の絵を同じようにCGENE.BASプログラムとして用意します．

ここまで準備ができたところで，“重ね合わせ”の開始です．前図の③を見てください．

まず背景の絵と透明の絵のAND(アンド)を取っています．こうすることにより，透明の絵の白い部分だけが透けて見え，背景の絵が見えます．キャラクタに相当する黒い部分のところは，影となって真っ黒になります．

次に，前図の④を見てください．いま，ANDを取って，キャラクタ部分だけが黒く抜けている背景の絵と，キャラクタの絵のOR(オア)を取っています．こうすることにより，抜けている部分にキャラクタの絵がスッポリと入り，見事，“本格的重ね合わせ”の完成です．

KASANE．BASプログラム

```
 CLS
 DIM PINK%(800)
     DEF SEG = VARSEG(PINK%(0))
     OFFSET% = VARPTR(PINK%(0))
     BLOAD "PINK.GRA", OFFSET%
     DEF SEG
 DIM GREEN%(800)
     DEF SEG = VARSEG(GREEN%(0))
     OFFSET% = VARPTR(GREEN%(0))
     BLOAD "GREEN.GRA", OFFSET%
     DEF SEG
 DIM TOUMEI%(800)
     DEF SEG = VARSEG(TOUMEI%(0))
     OFFSET% = VARPTR(TOUMEI%(0))
     BLOAD "TOUMEI.GRA", OFFSET%
     DEF SEG
 DIM DELICE%(800)
     DEF SEG = VARSEG(DELICE%(0))
     OFFSET% = VARPTR(DELICE%(0))
     BLOAD "DELICE.GRA", OFFSET%
     DEF SEG
 PUT (100, 1), TOUMEI%, PSET
 PUT (200, 1), DELICE%, PSET
 PUT (100, 50), PINK%, PSET
 PUT (200, 50), GREEN%, PSET
 PUT (100, 100), PINK%, PSET
 PUT (100, 100), TOUMEI%, AND
 PUT (200, 100), GREEN%, PSET
 PUT (200, 100), TOUMEI%, AND
 PUT (100, 150), PINK%, PSET
 PUT (100, 150), TOUMEI%, AND
 PUT (100, 150), DELICE%, OR
 PUT (200, 150), GREEN%, PSET
 PUT (200, 150), TOUMEI%, AND
 PUT (200, 150), DELICE%, OR
DO WIHLE 1
LOOP
```

★“本格的重ね合わせ”の原理

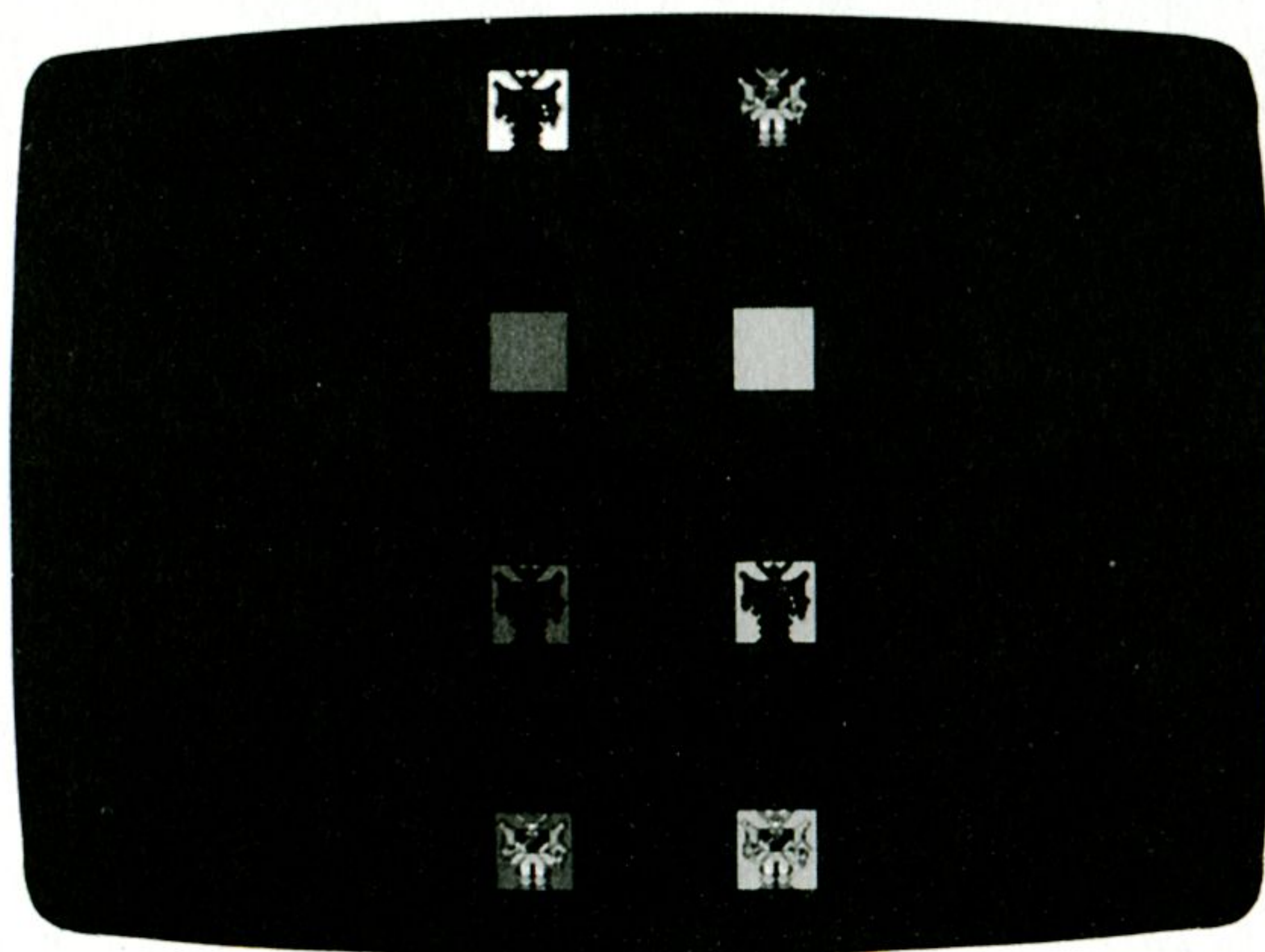
ANDを取ると，両方の絵で光っている部分のみが表示されます．つまり，一方でも真っ黒の部分は真っ黒になり抜けてしまいます．

ORを取ると，どちらか一方でも光っている部分が表示されます．これにより，両方の絵が合わさって表示されることになります．

★“本格的重ね合わせ”の特徴

“本格的重ね合わせ”の特徴としては，次のような点があげられます．

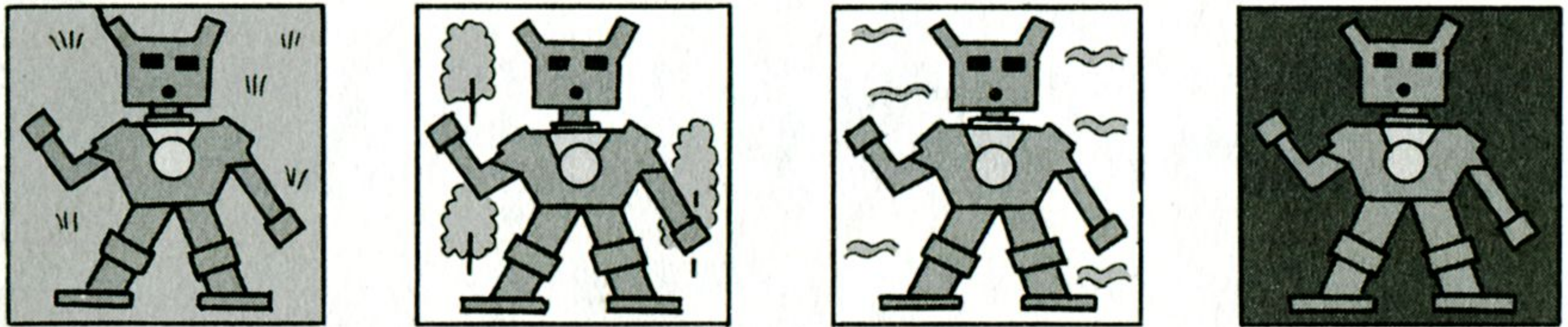
まず長所として，下の絵を選ばないため，背景と敵のキャラクタの“重ね合わせ”をやっておき，その上に味方のキャラクタを重ね合わせることができます．そして，遠景，近景と多重スクロールさせることが可能です．また，“重ね合わせ”の順番を変えることにより，木の向こう側をキャラクタが通過できるなど立体感が増します．

短所をあげると，“重ね合わせ”前の絵を持っておかなくてはいけません．また，プログラム処理に時間がかかります．

★カラー口絵
p.1参照

KASANE（本格的重ね合わせの原理）

★“超手抜き型重ね合わせ”

“本格的重ね合わせ”は上手に使えば，相当高度なことができます．でも，面倒なのも事実です．そこで，“超手抜き型重ね合わせ”をやってみましょう．

原理はいたって簡単です．“重ね合わせ”というのも若干気はずかしくなってしまいます．まず，下図を見てください．

超手抜き型重ね合わせ

やり方としては，もうあらかじめ“重ね合わせ”を行った絵を何種類も背景に合わせて用意しておく方法です．背景になにが来るかを読み取り，それに合わせたキャラクタの絵を表示します．

★“超手抜き型重ね合わせ”の特徴

このやり方ですと，毎回重ね合わせをする必要がありません．

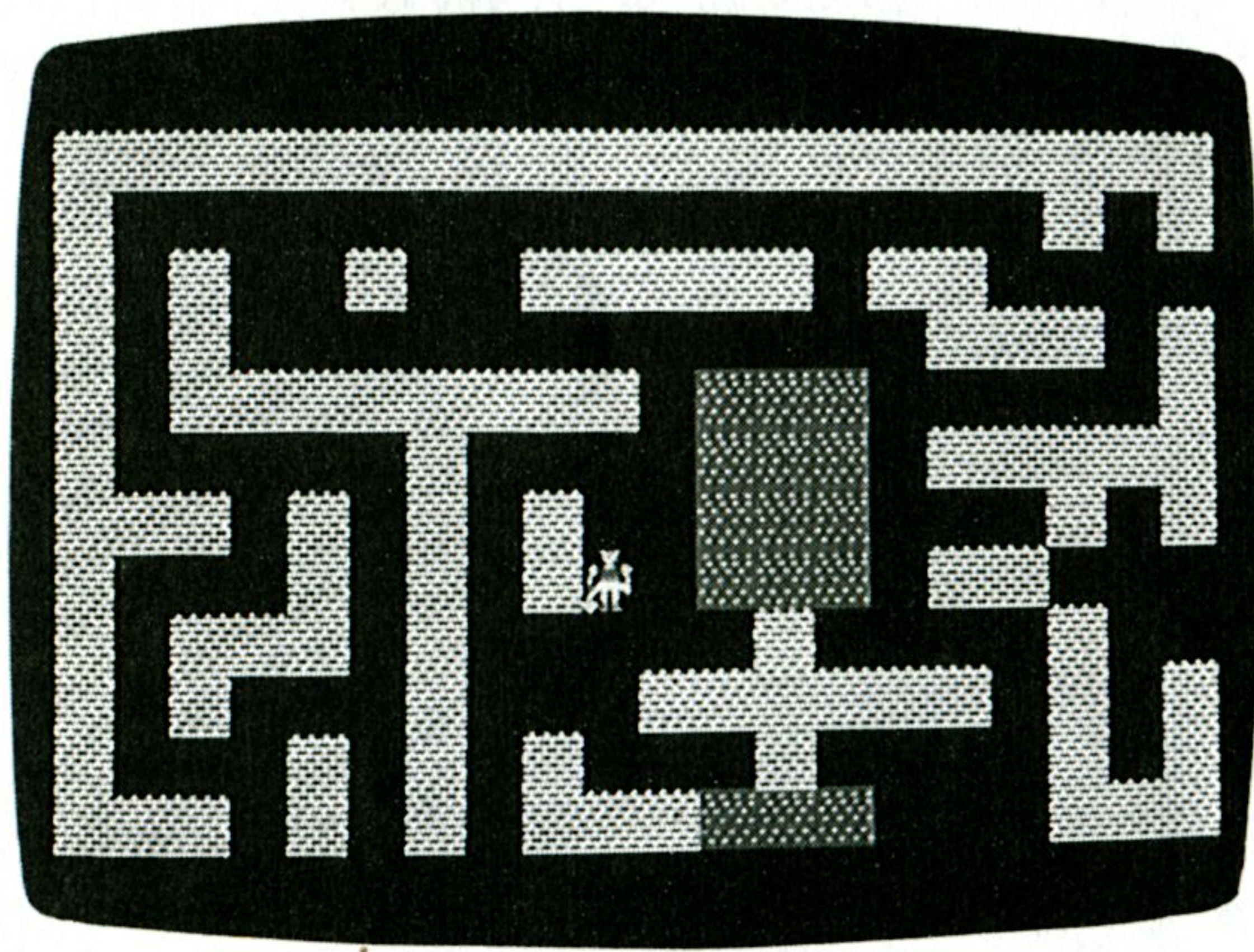
欠点として，あらかじめいろいろな場合の絵を作っておかなくてはなりません．

Quick BASIC

迷路の作成“MAZE”

ひところ，巨大迷路がはやりました．大きな迷路の中をみんなそろってウロウロと，なかなか楽しかったのですが，あちこちにでき過ぎて，あきられてしまったようです．

ゲームの中にも，『パックマン』．『ロードランナー』など，通れる所と通れない所のある迷路型のゲームが１つのジャンルを形作っています．今度は，この迷路に挑戦してみましょう．

MAZE（メーズ）とは，英語で迷路のことです．

『MAZE』(迷路ゲーム)

★ カラー口絵
p. 1参照

キャラクタがある方向に動けるかどうかというのは，一見簡単そうで，実は難かしい問題です．周りの色を見て，壁の色なら通れないようにするという方法などもありますが，ゲームの中には，通り抜けられる秘密の壁があったりして，そうは簡単にはいきません．また，ゲームの中には面の数を争っているものもあり，数百面などというゲームもあります．この面の絵すべてを絵で持っているとしたら，ハードディスクでさえ一杯になってしまい，とても普通のディスクなどでは収まりません．

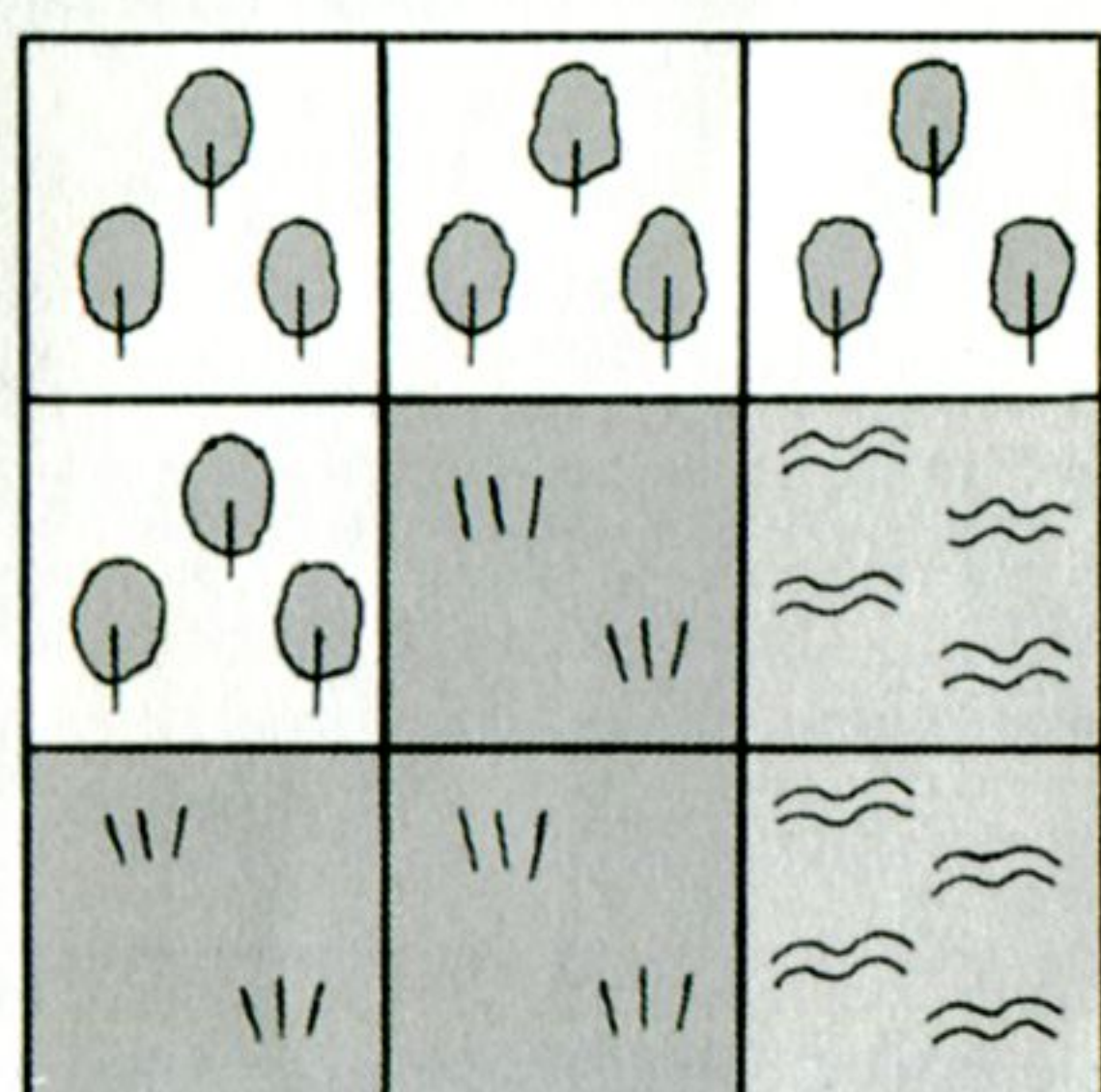
そこで出てくるのが，地形データという考え方です．

MAZEのプログラムに取りかかる前に，この地形データという考え方について見てみましょう．

★地形データ

すべての場面の絵を，全部絵のままの形で保存しようとすると，ディスクがいくらあっても足りません．また，その場所が通れるかどうかという判定も非常に面倒になってしまいます．そこで，ゲームでは地形データという考え方を使っています．

まず，画面の中を多数の小さな四角形，または六角形のあつまりに分割します．分割したあと，そのおのおのがどういう地形かをあらかじめ決めておいた数字や文字でデータとして持つようにします．

絵をかくときは，このデータを見て対応するマス目に必要な色を塗ったり，そこに対応するキャラクタをPUT（プット）したりします．

```
DATA_1,_1,_1
DATA_1,_0,_2
DATA_0,_0,_2
```

地形データ

地形データは上図のように，データの数字だけをその面のデータとして保存しておきます．画面に絵をかくときには，あらかじめ決めておいたルールに従って，1に対応する位置には森のキャラクタ，0には草原，2には水，というように，対応する適切な絵を表示します．

また，移動に際しては，たとえば，戦車の場合『移動しようとする方向のデータが2の場合は移動できない』というようにデータにより，移動できるかできないかを決めます．

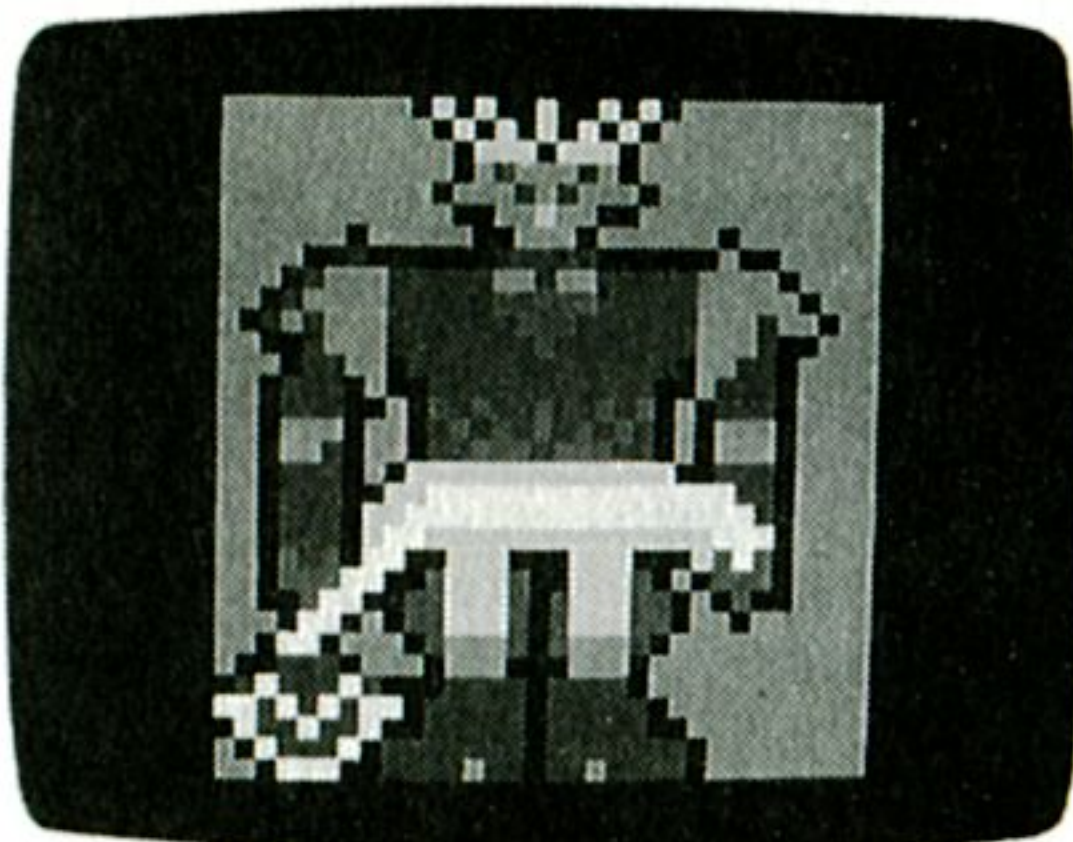
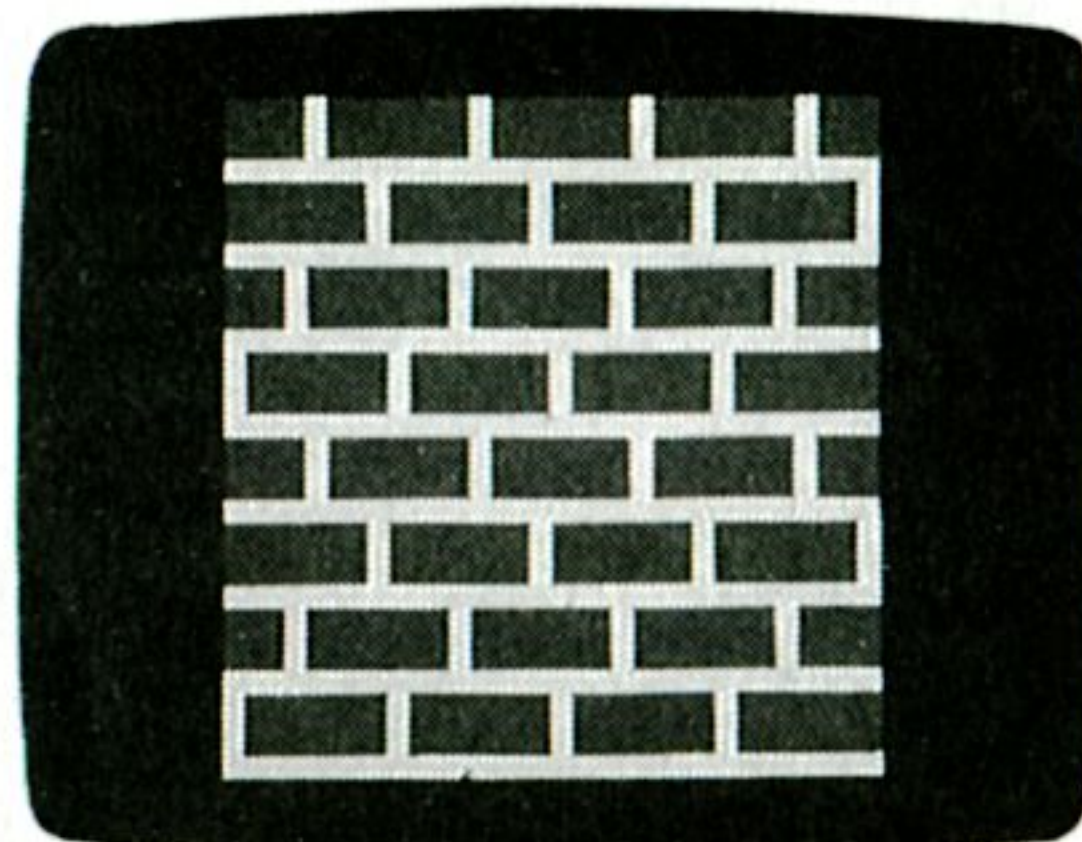
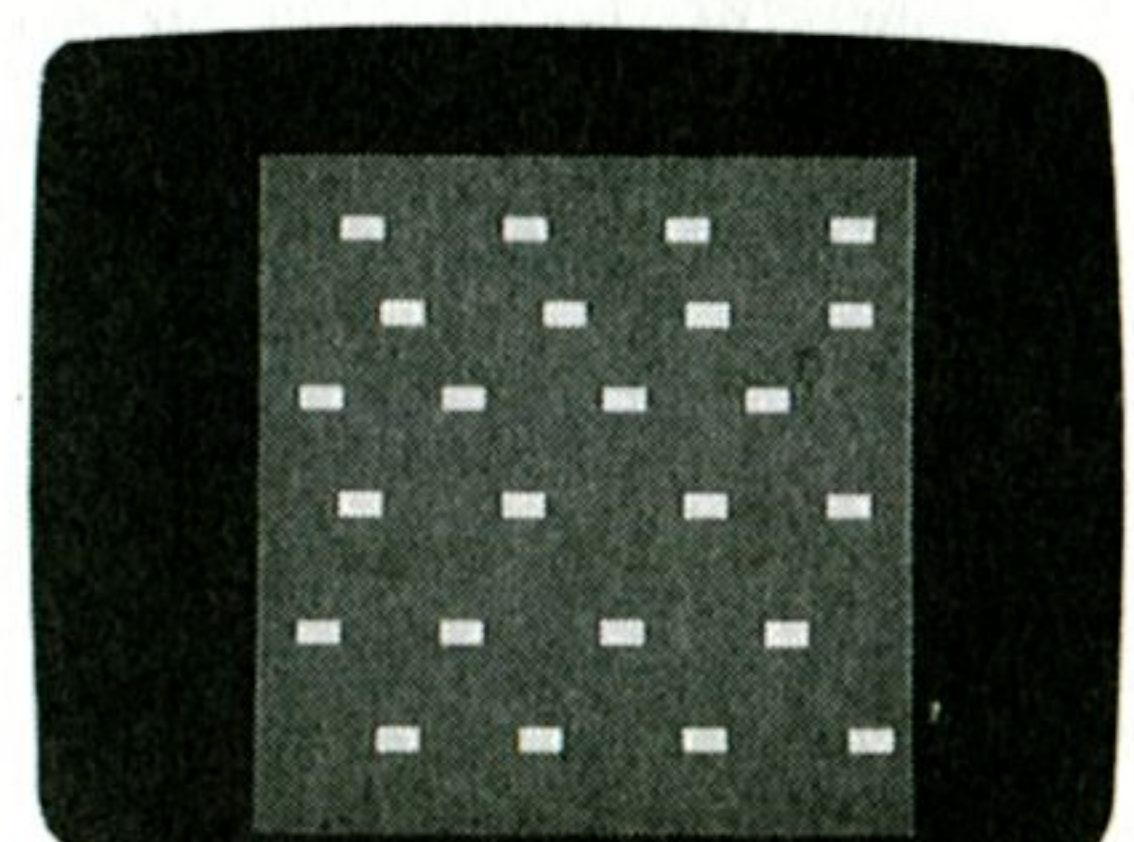
また，地形により移動で消費するエネルギーを変えることにより，道路なら遠くまでいけるが，森だと少ししか進めないというように使うこともできます．

さらに，データの数字を工夫して，たとえばデータを3としておくと，壁の絵は表示するけれど移動はできるなどとしておき，通り抜けできる秘密の迷路などというものも作れます．

このやり方だと，何十面，何百面という面や，広大なマップを持つ世界を作っても，必要なデータ量はかなり少なくて済みます．

地形データの考え方がわかったところで，それを応用したパソコンの中の巨大迷路 MAZE. BAS に挑戦してみましょう．

例によって主人公EXDELIC(エックスデリック)，レンガBRICK(ブリック)，水WATER(ウォーター)の絵が必要になります．あらかじめプログラム CGENE.BAS を使って作っておいてください．

★ カラーロ絵p. 2参照

主人公EXDELIC
(エックスデリック)

BRICK(レンガ壁)

WATER(水面)

名前さえ上の名前でセーブしてあれば，お姫様でも王子様でも狼男でも構いません．

MAZE.BAS 本体は131ページ以降にのっています．

プログラム入力の前には，13ページの注意をよく読んでから入力してください．

細かく説明すると，ページがいくらあっても足りません．そこで，ここではとくに大事なエッセンスについて説明します．

先程とりあげた地形データは，プログラムの最後にDATA の形で持っています．

MAZEデータ

```
DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
DATA 9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,9
DATA 9,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,5,0,0,0,5,9
DATA 9,5,0,5,0,0,5,0,0,5,5,5,5,5,0,5,5,0,0,0,0,0,5,0,5,9
DATA 9,5,0,5,0,0,0,0,0,0,0,0,0,0,0,0,5,5,5,0,5,0,5,0,5,9
DATA 9,5,0,5,5,5,5,5,5,5,5,0,6,6,6,0,0,0,0,0,5,0,5,0,5,9
DATA 9,5,0,0,0,0,0,5,0,0,0,0,6,6,6,0,5,5,5,5,5,0,0,0,5,9
DATA 9,5,5,5,0,5,0,5,0,5,0,0,6,6,6,0,0,0,5,0,5,5,5,0,5,9
DATA 9,5,0,0,0,5,0,5,0,5,0,0,6,6,6,0,5,5,0,0,0,5,0,5,5,9
DATA 9,5,0,5,5,5,0,5,0,0,0,0,0,5,0,0,0,0,5,0,0,5,0,0,0,9
DATA 9,5,0,5,0,0,0,5,0,0,0,5,5,5,5,5,5,0,2,0,5,5,0,5,0,9
DATA 9,5,0,0,0,5,0,5,0,5,0,0,0,5,0,0,0,0,5,0,5,0,0,5,0,9
DATA 9,5,5,5,0,5,0,5,0,5,5,5,6,6,6,0,0,0,5,2,5,5,0,0,5,9
DATA 9,5,0,0,0,5,0,5,0,0,0,0,6,6,6,0,0,0,5,0,0,0,5,0,5,9
DATA 9,5,0,5,5,5,5,5,0,0,5,0,0,5,0,0,5,5,5,5,0,0,5,0,5,9
DATA 9,5,0,5,0,0,0,5,0,0,5,0,5,5,0,2,0,0,0,0,0,0,5,0,5,9
DATA 9,5,0,0,0,5,0,5,0,5,2,0,5,0,0,5,5,5,5,5,0,0,5,0,5,9
DATA 9,5,5,5,0,0,0,5,0,5,0,5,5,0,0,5,0,0,0,0,5,5,5,0,5,9
DATA 9,5,5,5,5,2,5,0,5,0,0,0,6,6,6,0,0,5,0,0,0,5,0,0,5,9
DATA 9,5,0,0,0,0,0,0,5,5,5,0,6,6,6,0,0,5,0,0,0,0,5,0,5,9
DATA 9,5,0,5,5,5,5,5,0,5,0,5,0,5,0,0,0,5,5,5,5,5,5,0,5,9
DATA 9,5,0,5,0,0,0,5,0,5,0,5,5,5,5,5,2,5,0,0,0,0,0,0,5,9
DATA 9,5,0,0,0,5,0,0,0,5,0,5,0,0,0,5,0,5,0,5,5,2,5,5,5,9
DATA 9,5,0,5,5,5,5,5,0,5,0,0,5,5,0,5,0,0,0,5,0,0,0,0,5,9
DATA 9,5,0,5,0,0,0,5,0,5,0,0,0,5,0,5,5,5,5,5,0,5,5,0,5,9
DATA 9,5,0,0,0,5,5,5,0,5,0,5,2,5,0,0,0,0,0,0,0,0,5,0,5,9
DATA 9,5,5,0,0,5,0,0,0,0,5,0,0,0,5,5,0,5,5,5,0,0,0,5,9
DATA 9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,9
DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
```

単なる数字のら列に見えますが，0（ゼロ）が通り抜けのできる道，5がレンガ，6が水を表しています．周りを9で囲ってあるのは，一番端にきたときにそこから先のデータをチェックするためです．このデータがないとエラーを起こします．

このように，データはパソコンが許す限り大きくすることができます．ただし，必ず周りを通り抜けのできない値で囲むのを忘れないでください．

データの並び方を地形と同じ形にすると，ひとめで良くわかります．

データのままでは，扱い方が難しいので，配列に一度データをしまってやると，あとの取り扱いが楽になります．MAZE.BASプログラムでは，131ページの13行目以下の部分がデータを配列 MAZE に取り込むための部分です．

```
MAZEHAI．BASプログラム
  DIM MAZE(TATEMAX, YOKOMAX) AS INTEGER
  FOR I = 0 TO TATEMAX - 1
  FOR J = 0 TO YOKOMAX - 1
          READ MX
          MAZE%(I, J) = MX
  NEXT
  NEXT
```

次に，以下の部分で必要な絵を画面上にかいていきます．これは，画面を横20×縦12の領域に分け，現在の位置を中心に先程の配列の値を読みに行き，その値 FX をチェックするプログラムです．そして，FX の値が 5 なら対応する位置にレンガの絵をかきます．また，FX の値が 6 ならば，対応する位置に水の絵をかきます．

```
MAZEDRAW．BASプログラム
  FOR I = STARTI + 1 TO STARTI + 12
  FOR J = STARTJ + 1 TO STARTJ + 20
   FX = MAZE%(I, J)
   IF FX = 5 THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), BRICK%
   IF FX = 6 THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), WATER%
  NEXT
  NEXT
```

★迷路プログラムのキー入力

このキー入力ルーチンが，ある場所に移動できるかどうかの秘密です．いままでのキー入力ルーチンでは，数字キーを押すと，すぐにその方向に移動していました．ところが，このルーチンでは，その方向にいけるかどうかを確認して，いける場合にのみ移動します．

では，実際にその部分を見て見ましょう．MAZE.BAS の KEYCHECK: となっているサブルーチンの中を見てください．このサブルーチンの中の 4 行目で，

```
MAZEKEY.BASプログラム
  KEYCHECK:
  C$ = INKEY$
  IF C$ = "" THEN GOTO KEYCHECK
      IF C$ = "8" AND MAZE(B - 1, A) < 5 THEN A = A: B = B - 1: GOSUB MOVE
      IF C$ = "2" AND MAZE(B + 1, A) < 5 THEN A = A: B = B + 1: GOSUB MOVE
      IF C$ = "4" AND MAZE(B, A - 1) < 5 THEN A = A - 1: B = B: GOSUB MOVE
      IF C$ = "6" AND MAZE(B, A + 1) < 5 THEN A = A + 1: B = B: GOSUB MOVE
      IF C$ = "Q" THEN END
      IF C$ = "q" THEN END
      IF A = 24 THEN GOTO GOAL
  GOTO KEYCHECK
  END
```

IF_C$_=_"8"_AND_MAZE(B_-_1,_A)_<_5_THEN

となっています．このANDはその前の条件と，あとの条件が同時に満足されるときだけ，THEN以下の命令を実行しなさいという意味です．

つまり，上に移動するためには，まず8のキーが押されることが必要です．それと同時に，MAZE(B_-_1,_A)_<_5 が必要になります．Bは現在キャラクタのいる縦位置，Aは横位置を配列の中で示す値です．上に行くには，MAZE(B_-_1,_A)つまり，MAZEという配列の現在いる位置より1つ上に当たる部分の数字をチェックします．この数字が5より小さいときのみ，ANDの後ろ側の条件が成立します．つまり，キャラクタを上に動かすためには数字キーの8を押すのと，いまいるキャラクタの1つ上側の配列の値が4以下とが同時に必要です．ここで，5をレンガ，6を水というように5以上の数字に割り当てておけば，レンガや水の部分は通れません．

```
MAZE.BASプログラム
  SCREEN 88, 3, 1, 1
  CLS
  WINDOW SCREEN (0, 0)-(639, 399)
  DIM SHOUKYO%(800)
  GET (0, 0)-(31, 31), SHOUKYO%
  STARTJ = 0: STARTI = 2
  A = STARTJ + 10: B = STARTI + 6
  E = (A - STARTJ - 1): F = (B - STARTI - 1)
  PUT (C * 32, D * 32), SHOUKYO%, PSET
  C = E: D = F
  TATEMAX = 29
  YOKOMAX = 26
  DIM MAZE(TATEMAX, YOKOMAX) AS INTEGER
```

```
FOR I = 0 TO TATEMAX - 1
FOR J = 0 TO YOKOMAX - 1
        READ MX
        MAZE%(I, J) = MX
NEXT
NEXT
DIM BRICK%(800)
    DEF SEG = VARSEG(BRICK%(0))
    OFFSET% = VARPTR(BRICK%(0))
    BLOAD "BRICK.GRA", OFFSET%
    DEF SEG
DIM WATER%(800)
    DEF SEG = VARSEG(WATER%(0))
    OFFSET% = VARPTR(WATER%(0))
    BLOAD "WATER.GRA", OFFSET%
    DEF SEG
DIM KZ%(800)
    DEF SEG = VARSEG(KZ%(0))
    OFFSET% = VARPTR(KZ%(0))
    BLOAD "EXDELIC.GRA", OFFSET%
    DEF SEG
START:
CLS
IF STARTJ > YOKOMAX - 22 THEN STARTJ = YOKOMAX - 22
IF STARTJ < 4 THEN STARTJ = 0
IF STARTI > TATEMAX - 14 THEN STARTI = TATEMAX - 14
IF STARTI < 4 THEN STARTI = 0
FOR I = STARTI + 1 TO STARTI + 12
FOR J = STARTJ + 1 TO STARTJ + 20
   FX = MAZE%(I, J)
   IF FX = 2 THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), BRICK%
   IF FX = 5 THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), BRICK%
   IF FX = 6 THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), WATER%
NEXT
NEXT
E = (A - STARTJ - 1): F = (B - STARTI - 1)
C = E: D = F
PUT (C * 32, D * 32), SHOUKYO%, PSET
PUT (E * 32, F * 32), KZ%, PSET
KEYCHECK:
C$ = INKEY$
IF C$ = "" THEN GOTO KEYCHECK
    IF C$ = "8" AND MAZE(B - 1, A) < 5 THEN A = A: B = B - 1: GOSUB MOVE
    IF C$ = "2" AND MAZE(B + 1, A) < 5 THEN A = A: B = B + 1: GOSUB MOVE
    IF C$ = "4" AND MAZE(B, A - 1) < 5 THEN A = A - 1: B = B: GOSUB MOVE
    IF C$ = "6" AND MAZE(B, A + 1) < 5 THEN A = A + 1: B = B: GOSUB MOVE
    IF C$ = "Q" THEN END
    IF C$ = "q" THEN END
    IF A = 24 THEN GOTO GOAL
GOTO KEYCHECK
END
MOVE:
```

```
  IF A < 1 THEN A = 1
  IF A > YOKOMAX - 1 THEN A = YOKOMAX
  IF B < 1 THEN B = 1
  IF B > TATEMAX - 1 THEN B = TATEMAX
  E = (A - STARTJ - 1): F = (B - STARTI - 1)
  PUT (C * 32, D * 32), SHOUKYO%, PSET
  IF E > 16 AND (YOKOMAX - A) > 7 THEN STARTJ = STARTJ + 6: GOTO START
  IF E < 4 AND A > 7 THEN STARTJ = STARTJ - 5: GOTO START
  IF F > 10 AND (TATEMAX - B) > 4 THEN STARTI = STARTI + 4: GOTO START
  IF F < 2 AND B > 4 THEN STARTI = STARTI - 4: GOTO START
  IF E <= 0 THEN E = 0
  IF E >= 19 THEN E = 19
  IF F <= 0 THEN F = 0
  IF F >= 11 THEN F = 11
  PUT (E * 32, F * 32), KZ%, PSET
  C = E: D = F
  RETURN
  GOAL:
  CLS
  LOCATE 10, 35: PRINT "   GOAL    "
  LOCATE 11, 35: PRINT "CONGLATULATION"
  PUT (10 * 32, 10 * 32), KZ%, PSET
  END
  DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
  DATA 9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,9
  DATA 9,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,5,0,0,0,5,9
  DATA 9,5,0,5,0,0,5,0,0,5,5,5,5,5,0,5,5,0,0,0,0,5,0,5,9
  DATA 9,5,0,5,0,0,0,0,0,0,0,0,0,0,0,0,5,5,5,0,5,0,5,0,5,9
  DATA 9,5,0,5,5,5,5,5,5,5,5,0,6,6,6,0,0,0,0,5,0,5,0,5,9
  DATA 9,5,0,0,0,0,0,5,0,0,0,0,6,6,6,0,5,5,5,5,5,0,0,0,5,9
  DATA 9,5,5,5,0,5,0,5,0,5,0,0,6,6,6,0,0,0,5,0,5,5,5,0,5,9
  DATA 9,5,0,0,0,5,0,5,0,5,0,0,6,6,6,0,5,5,0,0,0,5,0,5,5,9
  DATA 9,5,0,5,5,5,0,5,0,0,0,0,0,5,0,0,0,0,5,0,0,5,0,0,0,9
  DATA 9,5,0,5,0,0,0,5,0,0,0,5,5,5,5,5,5,0,2,0,5,5,0,5,0,9
  DATA 9,5,0,0,0,5,0,5,0,5,0,0,0,5,0,0,0,0,5,0,5,0,0,5,0,9
  DATA 9,5,5,5,0,5,0,5,0,5,5,5,6,6,6,0,0,0,5,2,5,5,0,0,5,9
  DATA 9,5,0,0,0,5,0,5,0,0,0,0,6,6,6,0,0,0,5,0,0,0,5,0,5,9
  DATA 9,5,0,5,5,5,5,5,0,0,5,0,0,5,0,0,5,5,5,5,0,0,5,0,5,9
  DATA 9,5,0,5,0,0,0,5,0,0,5,0,5,5,0,2,0,0,0,0,0,0,5,0,5,9
  DATA 9,5,0,0,0,5,0,5,0,5,2,0,5,0,0,5,5,5,5,5,0,0,5,0,5,9
  DATA 9,5,5,5,0,0,0,5,0,5,0,5,5,0,0,5,0,0,0,0,5,5,5,0,5,9
  DATA 9,5,5,5,5,2,5,0,5,0,0,0,6,6,6,0,0,5,0,0,0,5,0,0,5,9
  DATA 9,5,0,0,0,0,0,0,5,5,5,0,6,6,6,0,0,5,0,0,0,0,5,0,5,9
  DATA 9,5,0,5,5,5,5,5,0,5,0,5,0,5,0,0,0,5,5,5,5,5,5,0,5,9
  DATA 9,5,0,5,0,0,0,5,0,5,0,5,5,5,5,5,2,5,0,0,0,0,0,5,9
  DATA 9,5,0,0,0,5,0,0,0,5,0,5,0,0,0,5,0,5,0,5,5,2,5,5,5,9
  DATA 9,5,0,5,5,5,5,5,0,5,0,0,5,5,0,5,0,0,0,5,0,0,0,0,5,9
  DATA 9,5,0,5,0,0,0,5,0,5,0,0,0,5,0,5,5,5,5,5,0,5,5,0,5,9
  DATA 9,5,0,0,0,5,5,5,0,5,0,5,2,5,0,0,0,0,0,0,0,0,5,0,5,9
  DATA 9,5,5,0,0,5,0,0,0,0,0,5,0,0,0,5,5,0,5,5,5,0,0,0,5,9
  DATA 9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,9
  DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
```

Quick BASIC

『デライス対ゲドバ』

★“ロールプレイングゲーム”

では，ここでキャラクタが，荒野や城の中を動きまわり，宝を探す“ロールプレイングゲーム”に挑戦してみましょう．

この中には，広大なマップの作り方，マップへのいろいろな物の表示，画面にウィンドウを作り，その中で処理を選択するやり方，荒野の中から城の中に入る場面の作り方，宝バコなど“ロールプレイングゲーム”に必要な要素がギッシリと詰まっています．あなたにとって，きっと役に立つことと思います．

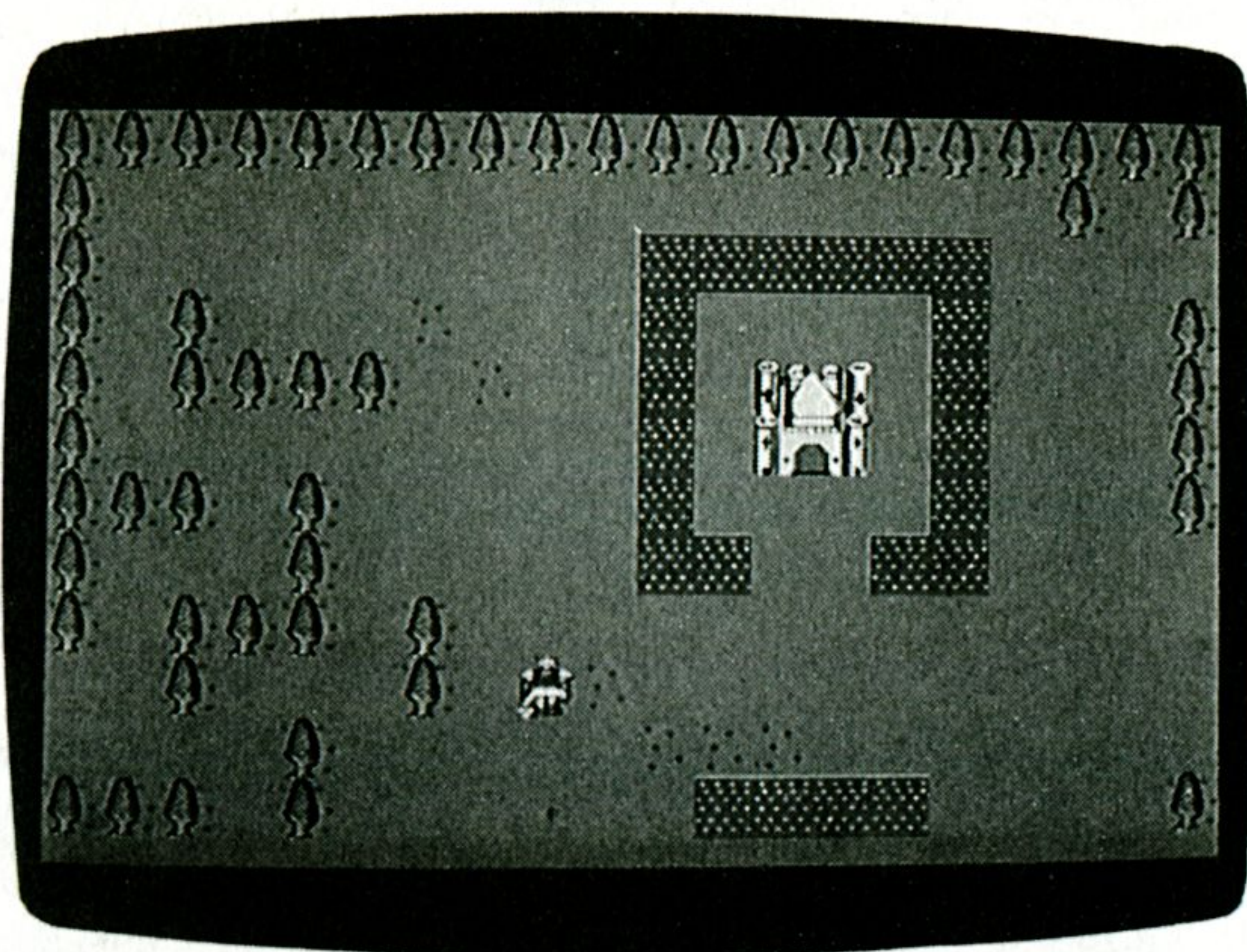
では，**「勇者よプログラムの中に旅立つが良い．」**

このプログラムは，大分本格的になってきただけに，いろいろなグラフィックスを多用しています．口絵１～４ページを参考に，CGENE.BASを使ってキャラクタをかいてください．同じ城なら城の絵であって，名前さえ同一名でセーブしてあれば，動作には支障がありません．あなた自身の世界を作り上げてください．

絵をかき終わったら，145ページからのDLVSGD.BASのプログラムを入力し，保存，実行してみてください．

プログラム入力の前に13ページの注意をよく読んでから入力してください．

このプログラムでの物語は，ゲームとしては未完成です(ゲームとして完結させると，あと何十ページもリストが増えてしまいます)．

ただし，ウィンドウの作り方，フラッグの立て方，ウィンドウを使った命令，広大なマップの作り方，他の場所への瞬間移動などとおよそ“ロールプレイングゲーム”として必要な機能はすべて盛り込んであります．

★カラー口絵
p. 2参照

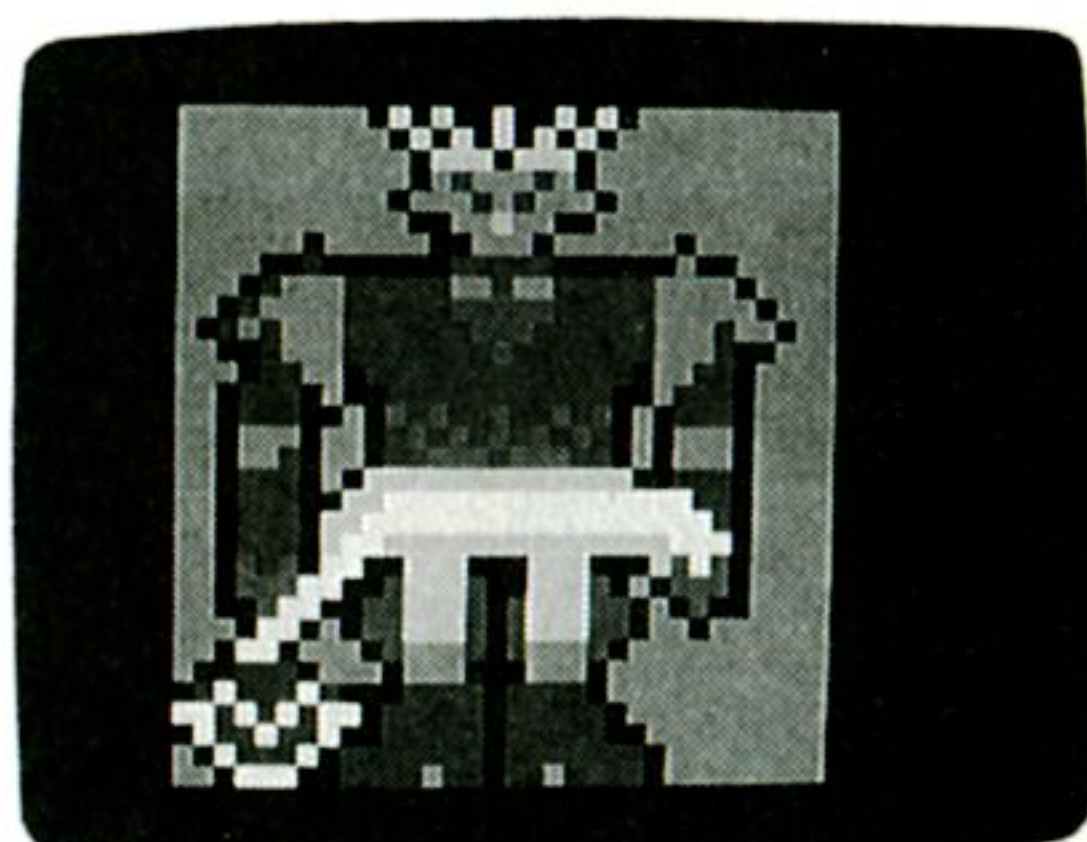
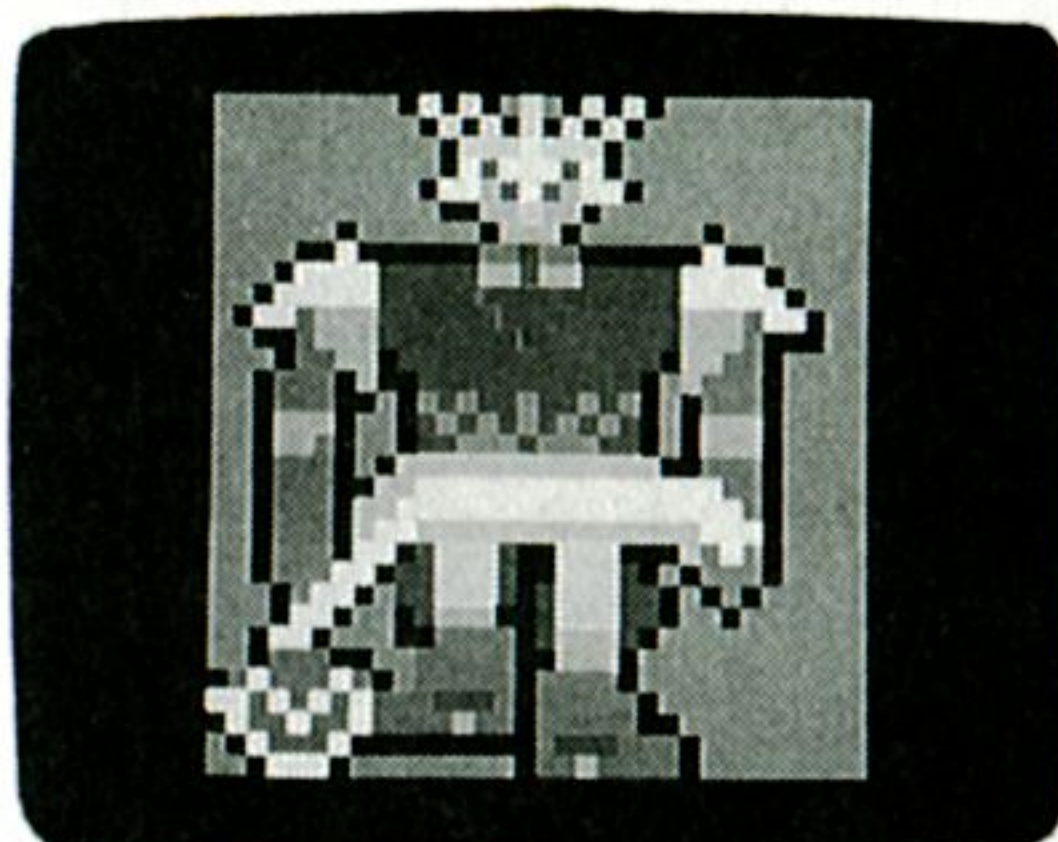
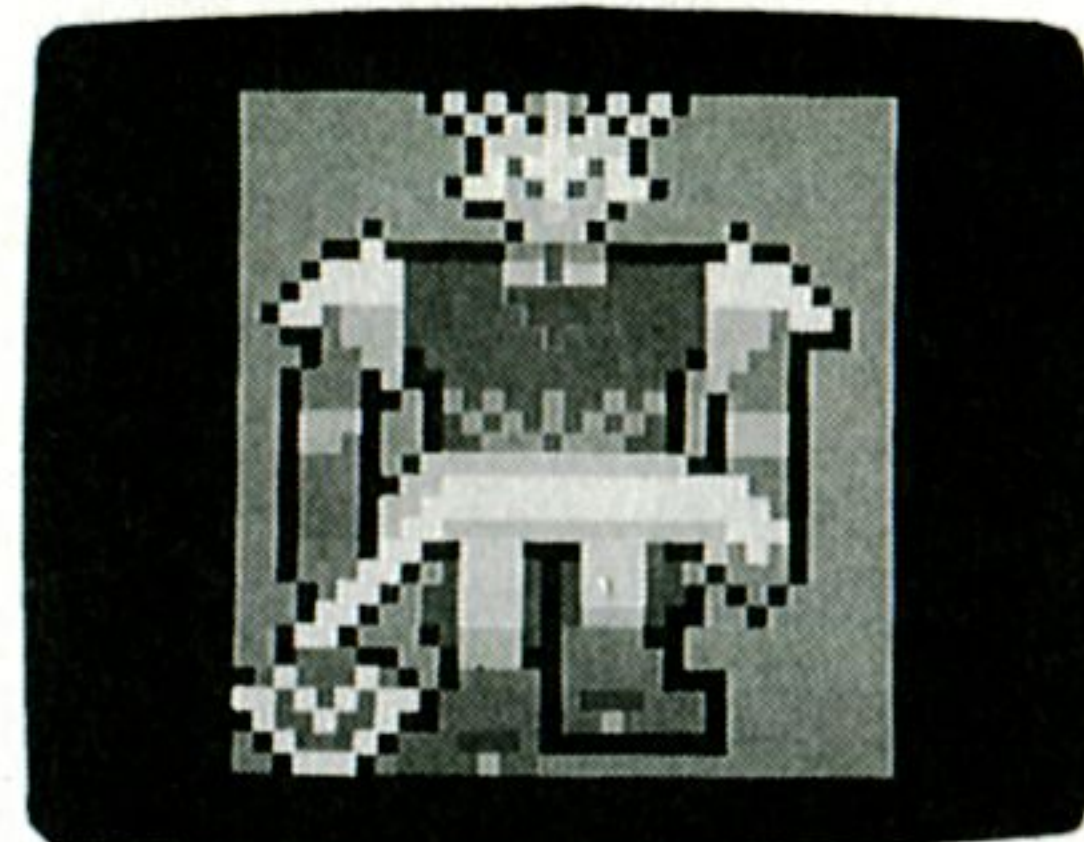
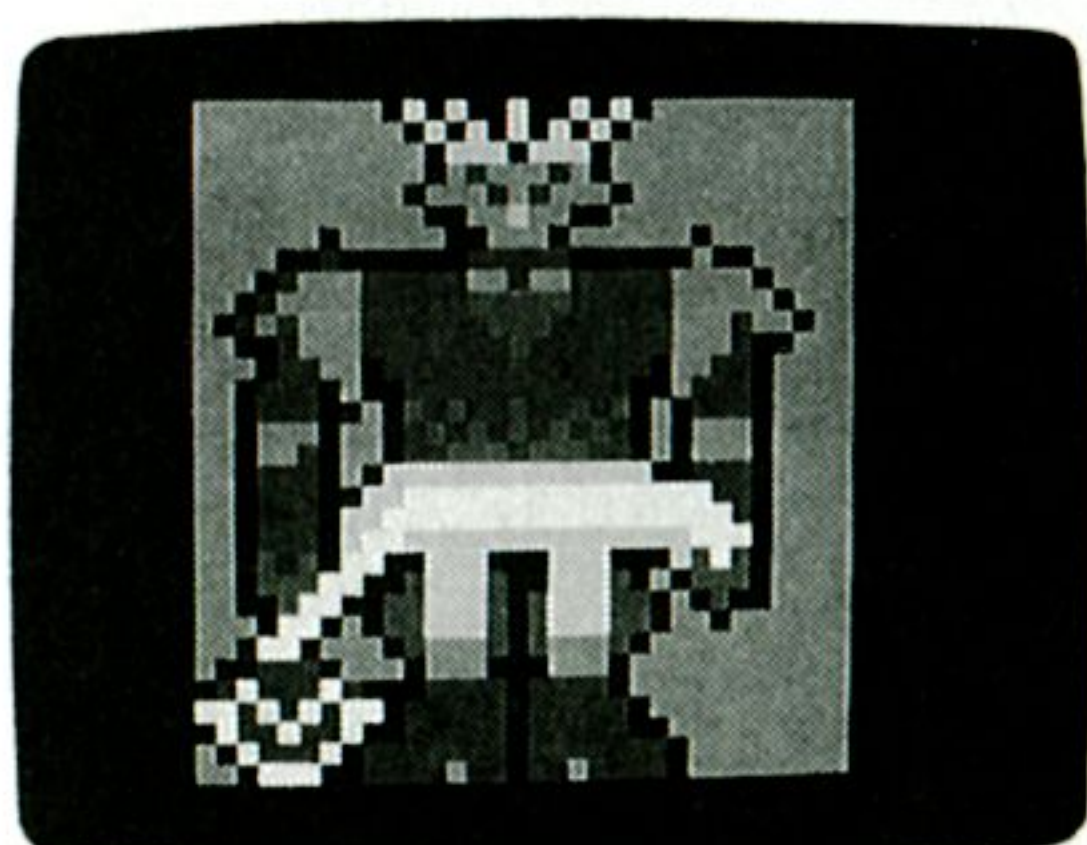
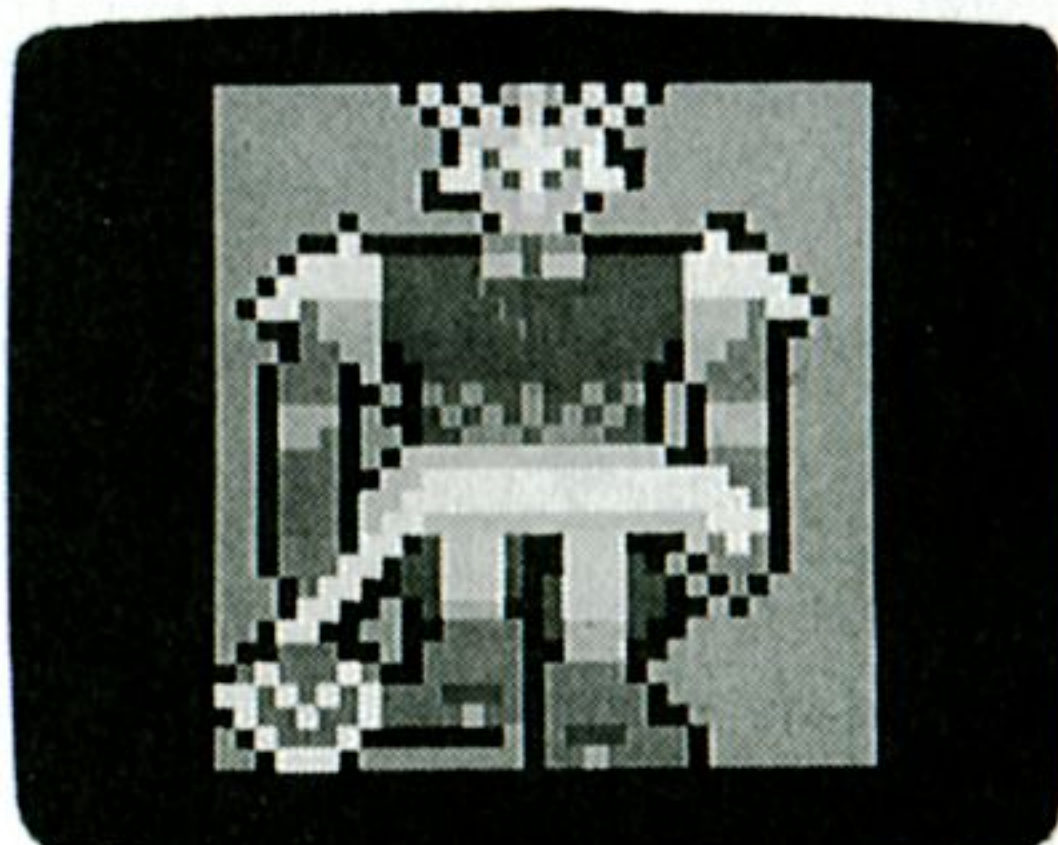
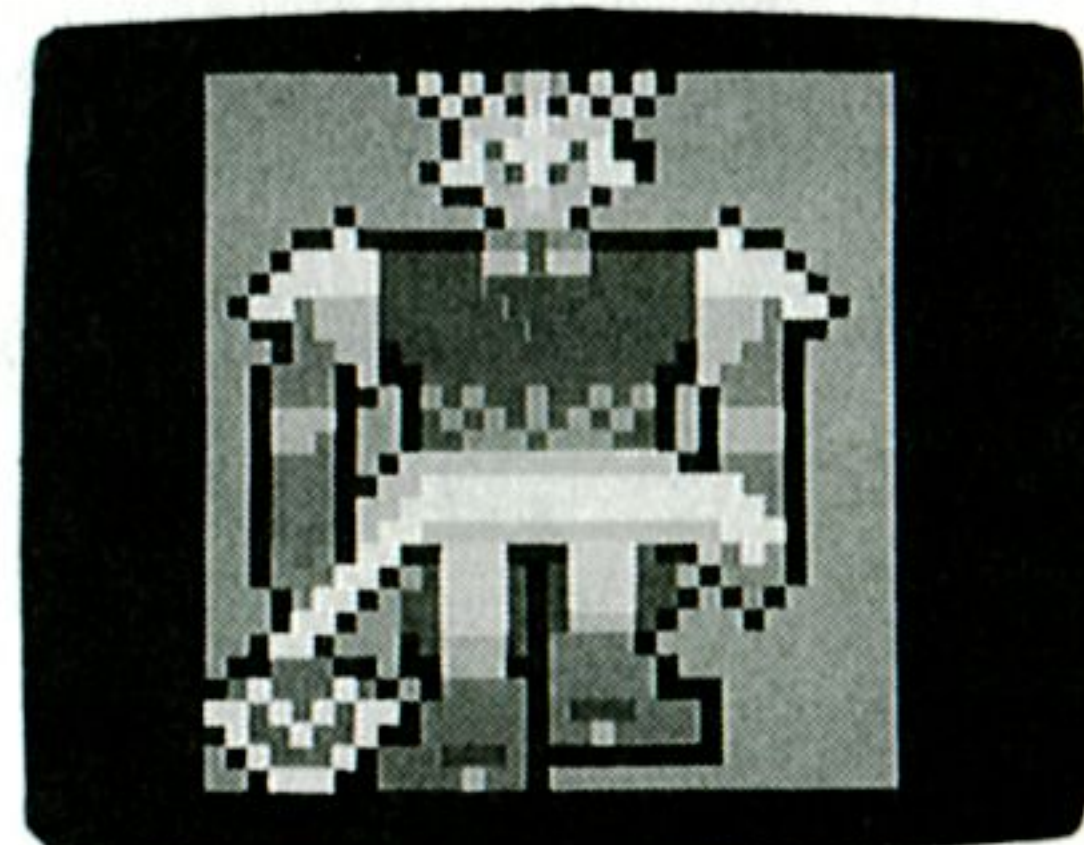
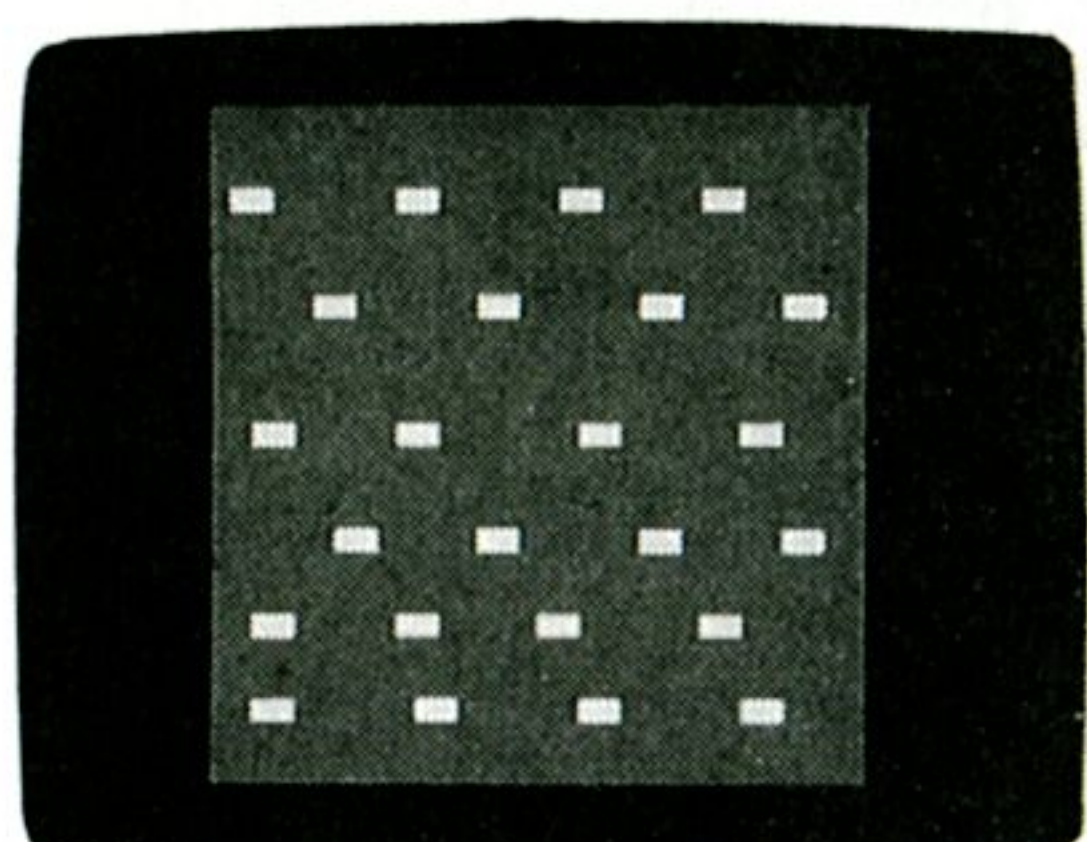
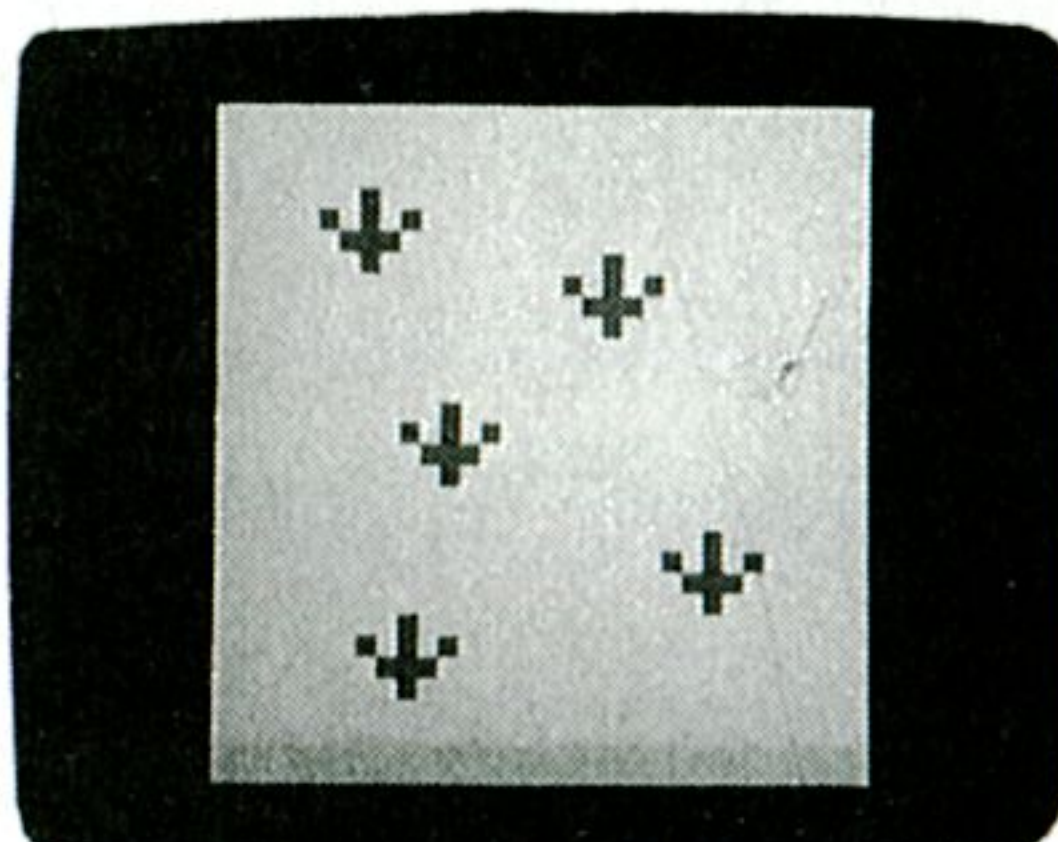
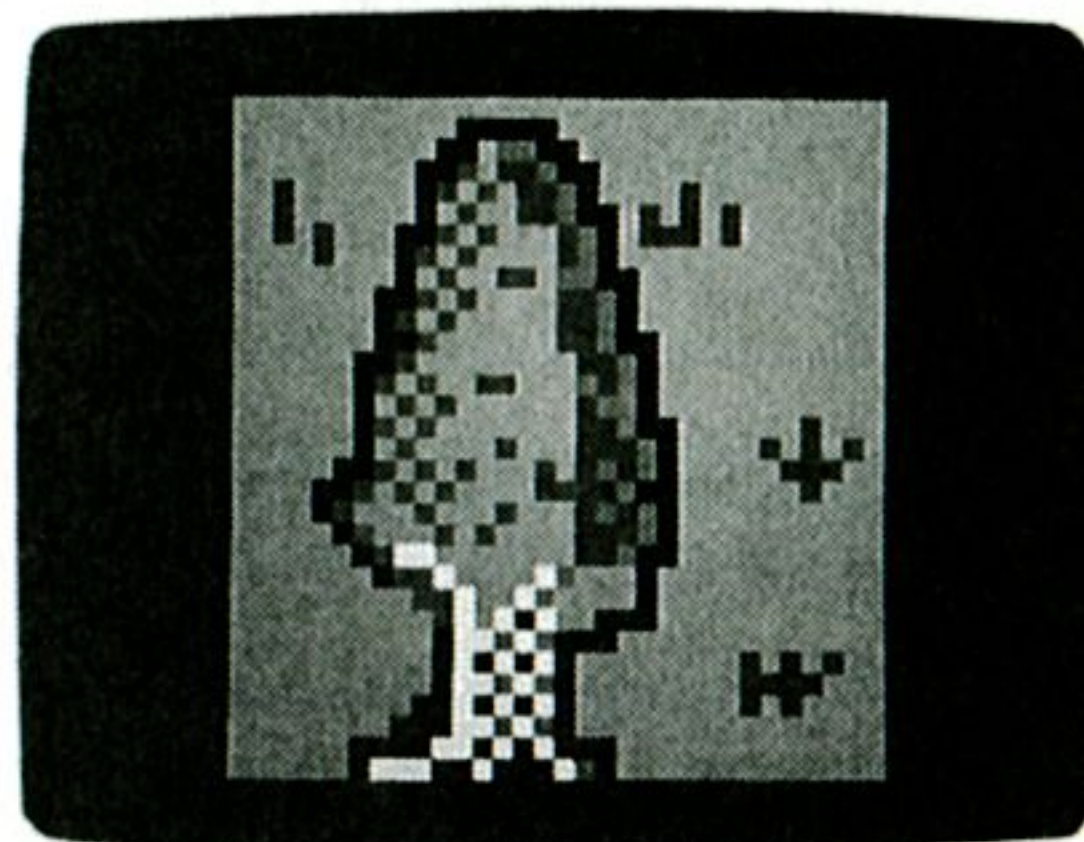
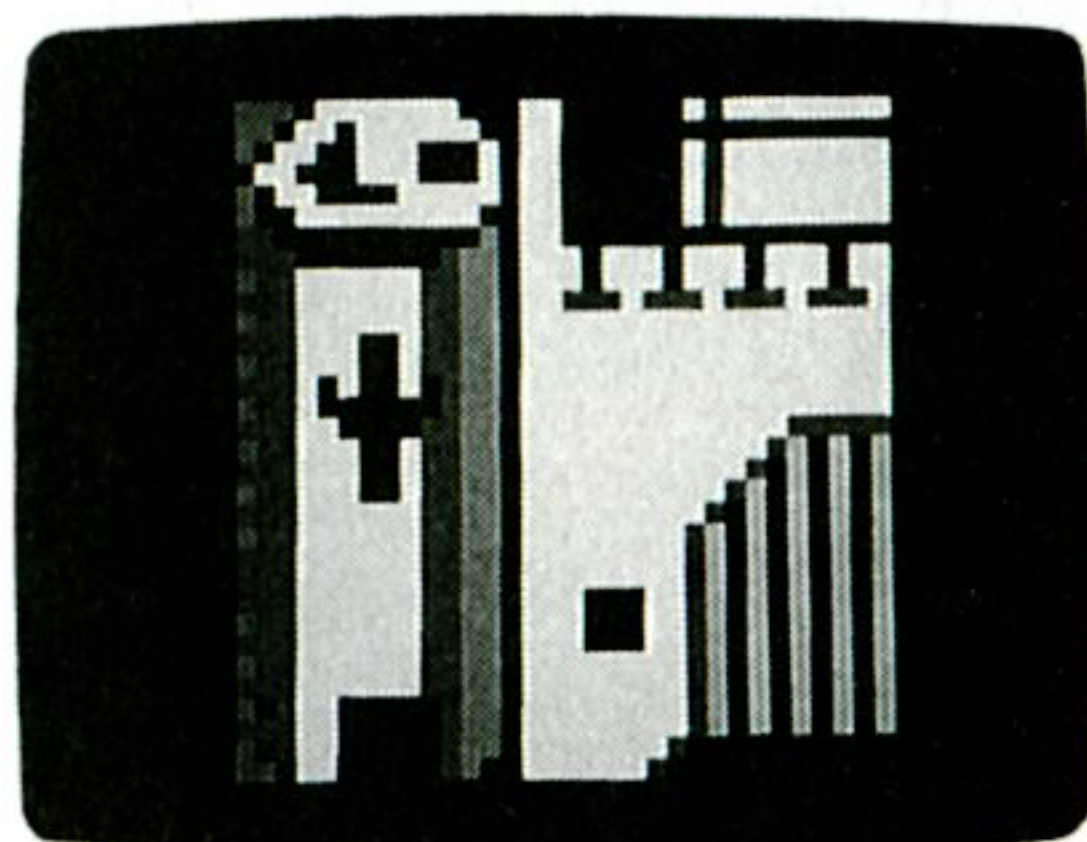
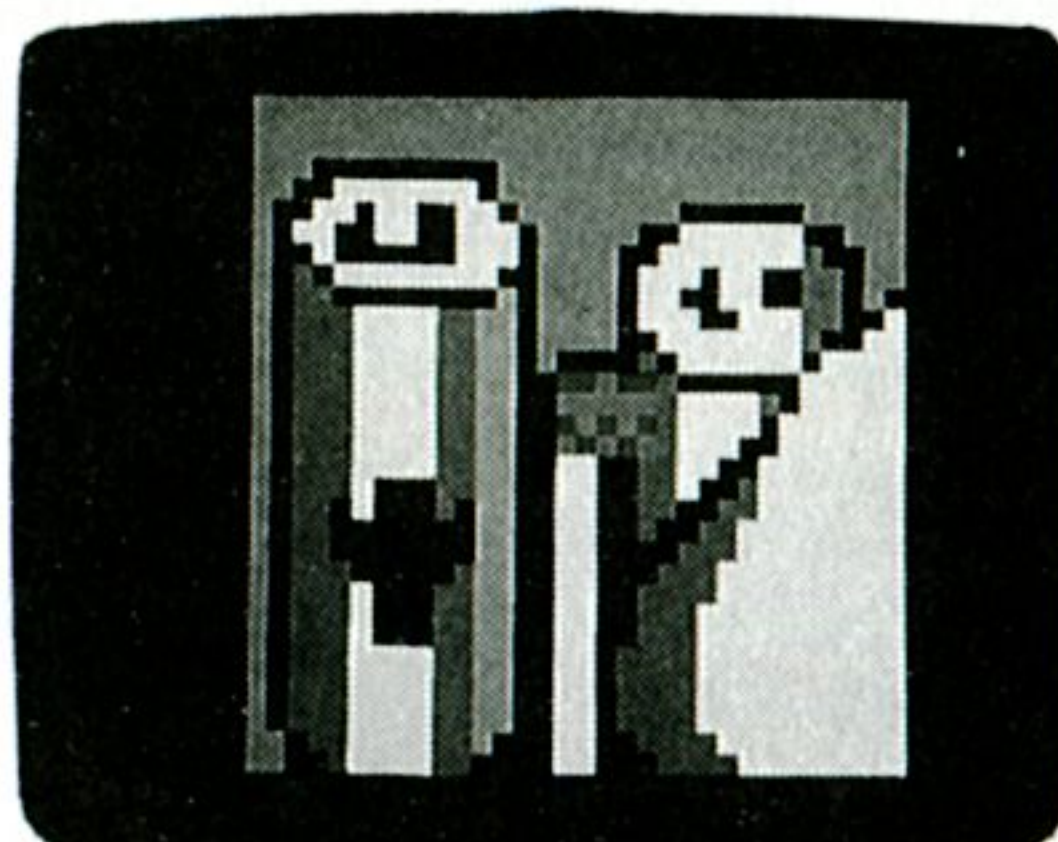
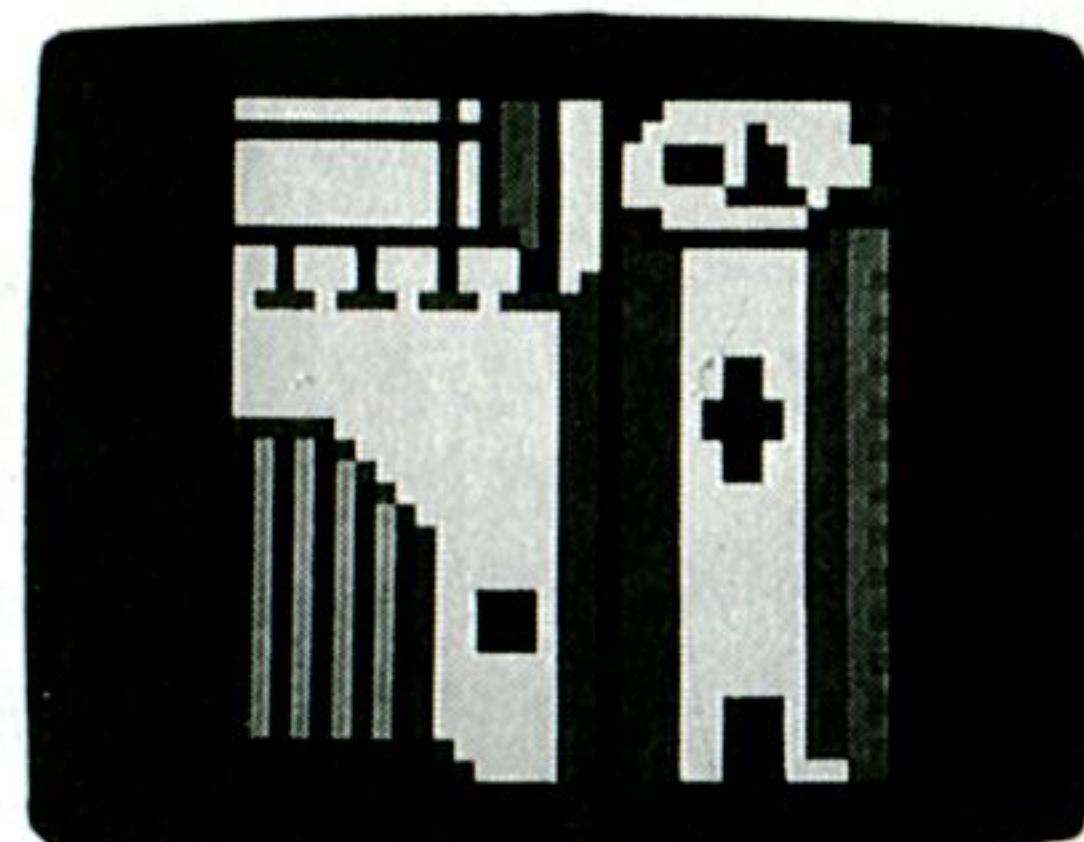
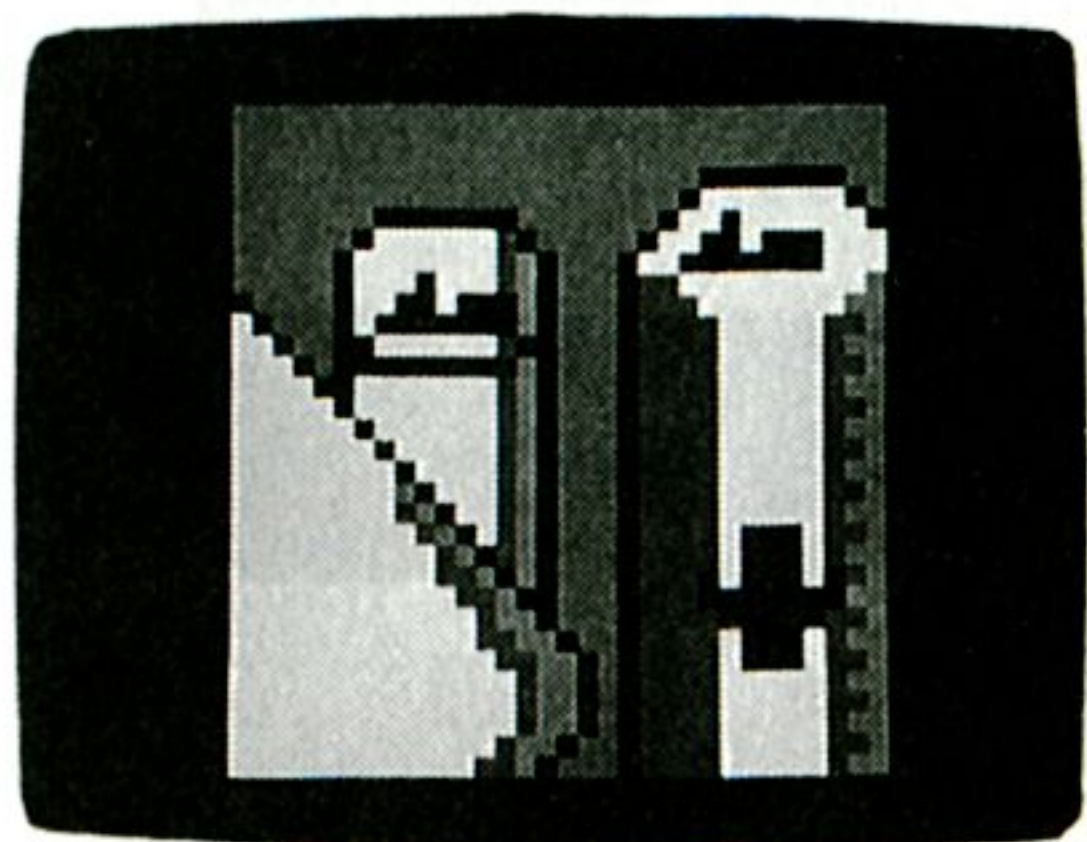
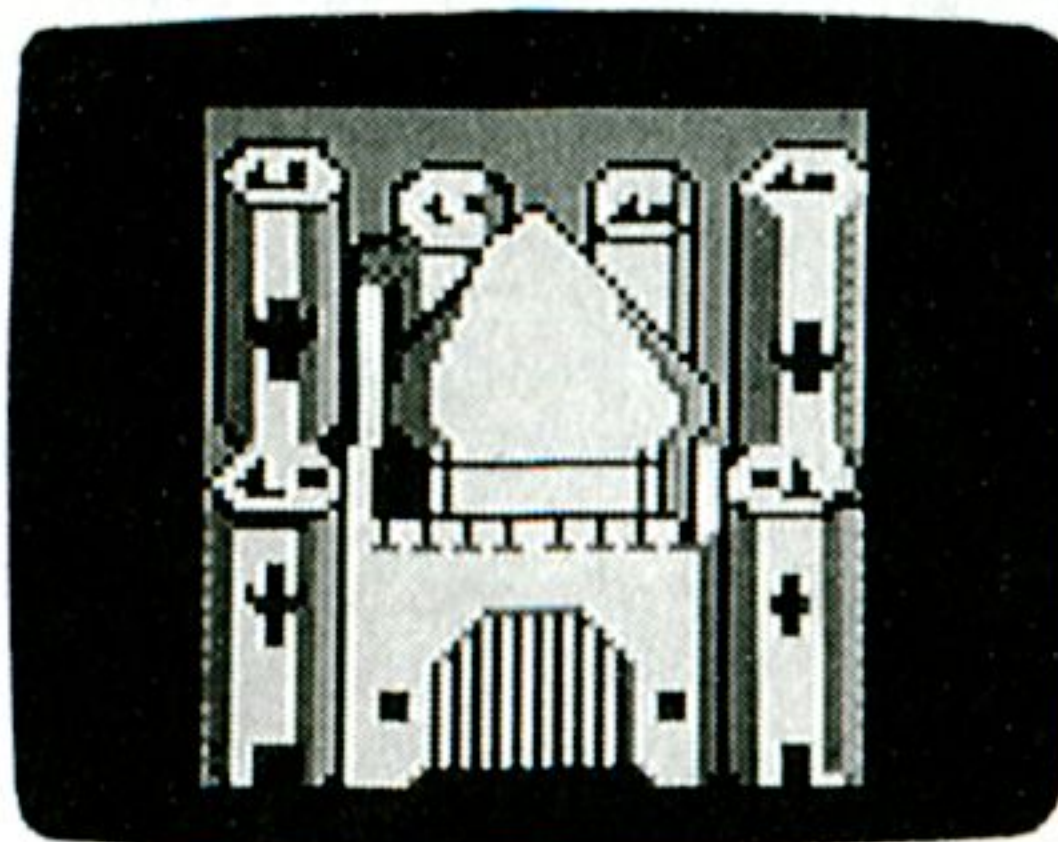
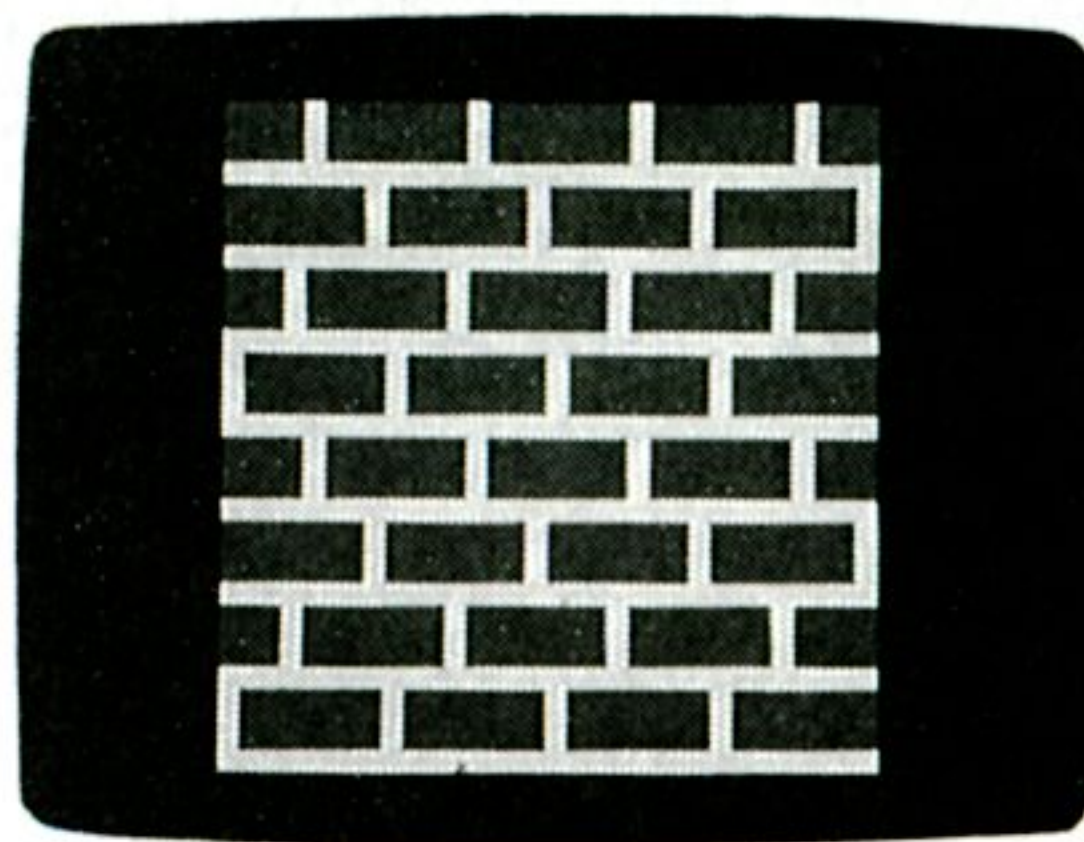
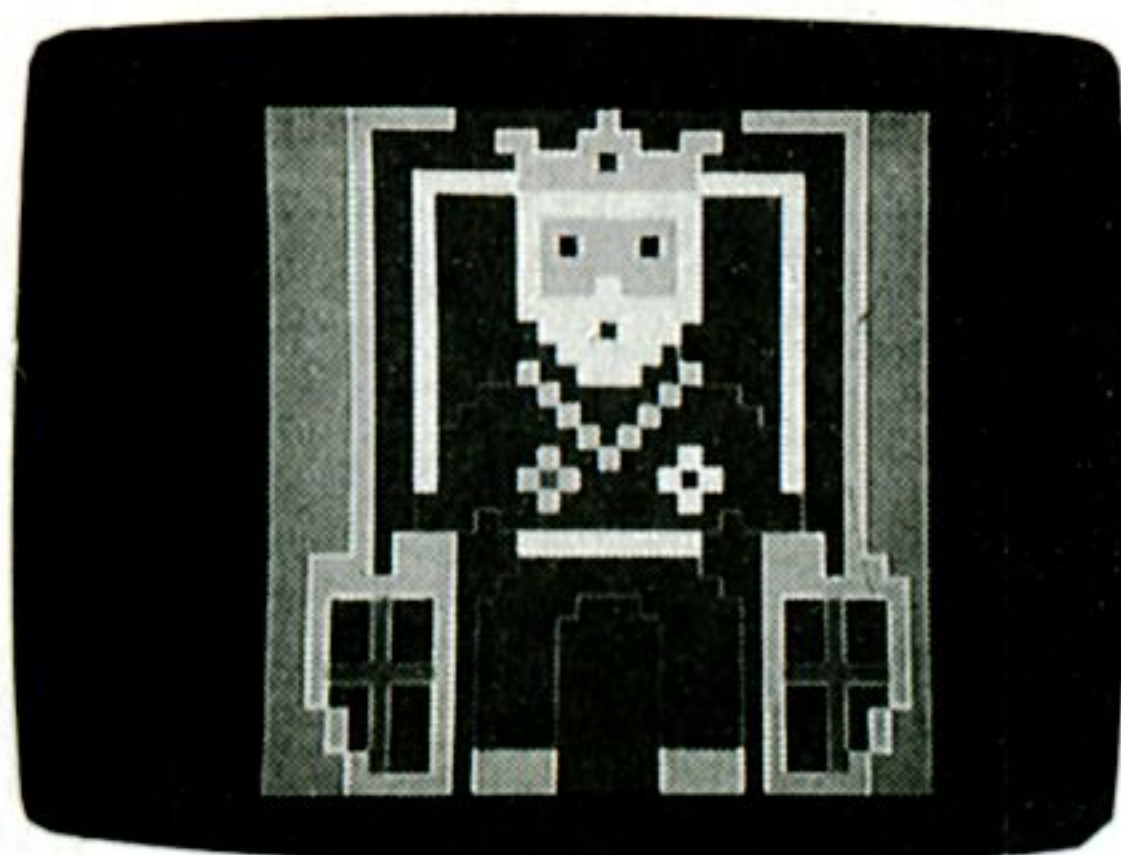
『デライス対ゲドバ』(ロールプレイングゲーム)

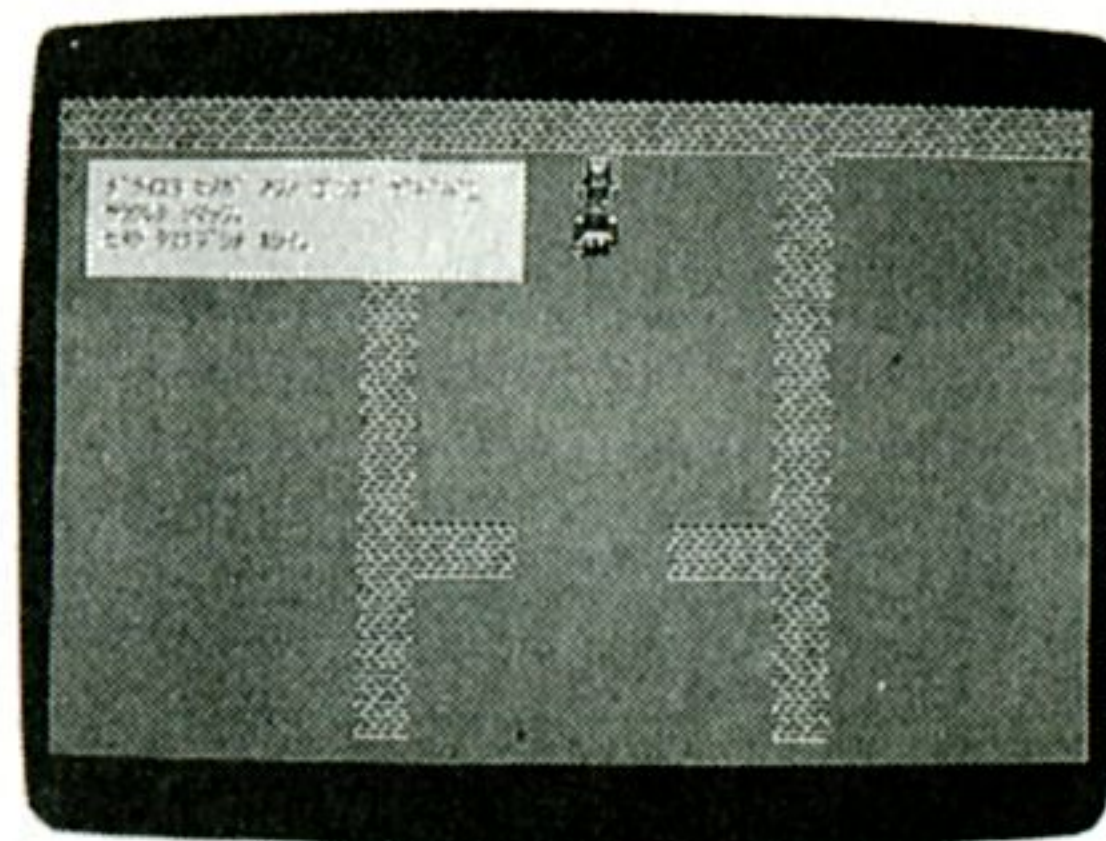
あとは，あなた自身のアイデアで，マップや敵を作り，世界中に1つしかないあなた自身のゲームを作ってください．友達に見せて，「俺が作ったんだよ」といえば尊敬されることは間違いありません．

145ページからのDLVSGD.BASのプログラムを見てください．このプログラムの位置と周囲の表示の部分は，前に書いたMAZE.BASのプログラムと良く似ていますが，1つだけ大きく違うところがあります．まず，145ページ20行目のDIMの部分ですが，配列の要素を数字ではなくストリング，つまり文字として宣言しています．

148ページの14行目以下の部分においても，FX$と文字として読み出し，また，IF_FX$_=_"1"　と，文字として比較しています．

```
DLYOMIDA．BASプログラム
FOR I = STARTI + 1 TO STARTI + 12
FOR J = STARTJ + 1 TO STARTJ + 20
      FX$ = MAZE$(I, J)
IF FX$ = "1" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), GRASS%, PSET
IF FX$ = "4" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), WATER%, PSET
IF FX$ = "5" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), TREE%, PSET
IF FX$ = "6" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLLB%, PSET
IF FX$ = "7" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLLT%, PSET
IF FX$ = "8" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLRT%, PSET
IF FX$ = "9" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLRB%, PSET
IF FX$ = "A" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), BRICK%, PSET
IF FX$ = "K" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), KING%, PSET
NEXT
NEXT
```

★カラー口絵 p. 2～3参照

GREENSS（緑地に直立）　GREENSR（緑地，右足上げ）　GREENSL（緑地，左足上げ）

PINKSS（ピンク地に直立）　PINKSR（ピンク右足上げ）　PINKSL（ピンク左足上げ）

WATER（水）　GRASS（草原）　TREE（木）

CASTLLB（城，左下）　CASTLLT（城，左上）　CASTLRB（城，右下）

CASTLRT（城，右上）　城（4つを組み合わせると）　BRICK（レンガ壁）

★カラー口絵p. 3参照

KING（王様）

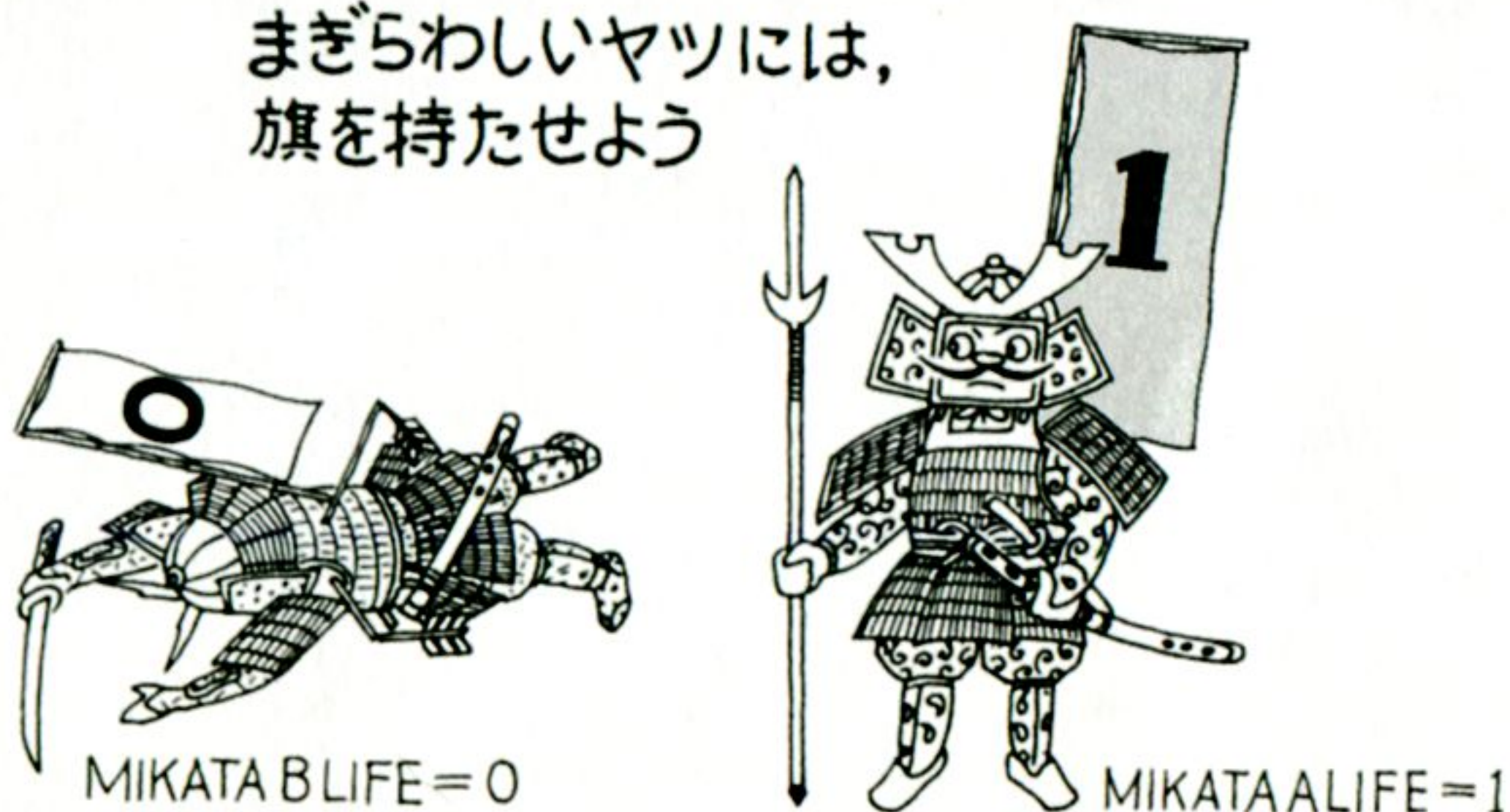
城中の王様とデライス

こうやって，文字として比較することにより，0～9までの数字で示すよりはるかに多くの地形や人，物などを表示させることができます．アルファベットの文字は，大文字，小文字の区別がありますし，その他，カタカナ，グラフィック文字など，表示できる文字の数だけ区別することができます．2桁，3桁の数字で表しても良いのですが，データ文を見たときに，ひとめで何になるのかがわかりにくくなります．また，広いマップを作るときに，横方向のデータが1行に収まりきれなくなる可能性があります．そうなると，マップに当たるデータ部分を見たとき，非常にわかりにくくなります．

「ムム，あの旗は」
「ハハー，楠木正成にございます」
（まぎらわしいやつには旗を持たせよう）

昔の戦いでは，みな自分の軍団に旗を持たせました．同じような姿で戦っているので，目印がなければ敵か味方かわかりません．また，あとで恩賞のさたのときにも，自分が活躍したのをアピールしておかなくてはなりません．そこで，鎧（よろい）を真赤にしたり，旗を立てたり，いろいろと工夫しました．有名になると，見ただけで敵がビビッて逃げ出したりする効果もあったようです．

まぎらわしいヤツには，
旗を持たせよう

0
MIKATA B LIFE＝0

1
MIKATA A LIFE＝1

プログラムの世界でも，この旗が活躍するのです．いろいろな事象がどうなっていたかを管理するのは，プログラムが複雑になると非常に難しくなります．そこで，ある事象が発生したら，ある変数の値を変えて旗を上げさせます．

次に，その事象が起きているかどうかを判定する必要ができたときには，その旗が上がっているかどうかをチェックしてやれば良いわけです．

このやり方は，とくに事象の発生と，それに依存するイベントがプログラムの中で非常に離れたところで起こるときに，とくに有効です．

しかし，実際に旗を上げるわけにはいきませんから，プログラムの中では，次のように処理をします．

まず，初期設定である変数を0または1にしておきます．ある事象が起きたら，そのときに0を1に，または1を0に変えます．チェックするルーチンでは，その変数が1か0かを見て，その事象が起きたのかどうかを調べてそれに応じた処理を行います．

では，この旗(フラッグ)の原理を使った移動ルーチンについて見てみましょう．149ページ下から3行目から150ページ19行目まで，以下の部分を見てください．

CHANGEDL．BASプログラム

```
IF B >= 4 AND B <= 7 AND A >= 12 AND A <= 15 AND TOWN = 0 THEN
FOR I = 0 TO TATEMAX - 1
FOR J = 0 TO YOKOMAX - 1
        MAZE$(I, J) = MACHIA$(I, J)
NEXT
NEXT
TOWN = 1
STARTI = 22
STARTJ = 5
OLDA = STARTJ + 1
OLDB = STARTI + 1
A = 16
B = 26
FOR I = 0 TO 800
EXDESR%(I) = PINKSR%(I)
NEXT I
FOR I = 0 TO 800
EXDESL%(I) = PINKSL%(I)
NEXT I
FLOORCOLOR = 5
GOTO START
END IF
```

ここでは，TOWN＝0という旗(FLAG)が出てきます．最初に初期設定で，TOWNの値は0にセットしてあります．これは，主人公『デライス』が，町の外にいることを示しています．つまり，主人公が外にいて，AとBの値があらかじめ定められた値である城の前の位置になる場所に移動して来たとき，THEN以下の部分に移ります．THEN以下の部分では，　MAZE$(I,_J)　という配列の中のすべての要素を，　MACHIA$(I,_J)　という配列と入れ替えています．つまり，こうすることで，地形の元となるデータを全部取り替えてしまいます．

その後，GOTO_START　という部分で最初の書き替えルーチンに飛ばして，絵を全面的にかき直しています．

この方法を使えば，お城の中の階段のところに来たら2階に移動するとか，洞くつの入口に立つと，地下の大洞くつに入ることが簡単にできます．また，世界のハジの方に立つと，また別の世界というようにメモリの許す限り，マップを大きくすることができますし，データをシーケンシャルファイルにしてディスクにしまっておき，それを必要に応じて読むようにすれば，ディスクの容量一杯まで広大な世界を作ることもできます．

★宝箱メッケ

徳川家の残した埋蔵金数万両，カリブ海に沈んだ海賊船，宝には男のロマンをくすぐるものがあります．中には一生，宝伝説につかれて山を掘り続けたりする人もいますが，本当にカリブ海で沈んだ船を引き上げ，莫大な財宝を手にした人もいます．冒険とロマンの世界です．

ゲームの世界でも，必要なアイテムを手に入れたり，ゴールドが増えたりと，宝箱は重要な役割を持っています．

では，宝探しに出かけましょう．

まず，150ページの下から12行目の部分です．

```
TAKARA1.BASプログラム
  IF TAKARA1 = 0 AND TOWN = 0 THEN GOSUB TAKARACHECK
```

同じところにいって，何回でも宝が見付かるのでは不自然です．まず最初に，TAKARA1_=_0　として設定しておきます．ここで宝1が捕られるまでは変数TAKARA1の値は0です．そこで，TAKARA1_=_0　で，かつデライスが外にいる，つまりTOWN変数が0のときにだけ，TAKARACHECKのサブルーチンに飛んで行かせます．

では，サブルーチンTAKARACHECKでは何をやっているのでしょう．151ページの下から21行目以下の部分です．

TAKARA2．BASプログラム

```
TAKARACHECK:
IF A = 10 AND B = 10 THEN
PUT ((A - STARTJ) * 32, (B - STARTI - 1) * 32), TAKARAC%, PSET
GET (10, 20)-(80, 150), TEMPO%
GET (200, 40)-(470, 110), TEMPO2%
LINE (10, 20)-(80, 150), 0, BF
LINE (10, 20)-(80, 150), 7, BF
LINE (11, 21)-(79, 149), 0, BF
CLS 2
LOCATE 3, 5: PRINT "OPEN"
LOCATE 4, 5: PRINT "TAKE"
LOCATE 5, 5: PRINT "USE"
LOCATE 6, 5: PRINT "END"
DO WHILE 1
WINDOW2:
   LOCATE ARROW, 3
   PRINT "*"
WINDOWKEY$ = INKEY$
IF WINDOWKEY$ = "" THEN GOTO WINDOW2
WINDOWKEY1 = ASC(LEFT$(WINDOWKEY$, 1))
WINDOWKEY2 = ASC(RIGHT$(WINDOWKEY$, 1))
   IF WINDOWKEY1 = 0 AND WINDOWKEY2 = 72 THEN
   ARROW = ARROW - 1: OLDARROW = ARROW + 1
   END IF
   IF WINDOWKEY1 = 0 AND WINDOWKEY2 = 80 THEN
   ARROW = ARROW + 1: OLDARROW = ARROW - 1
   END IF
   IF ARROW >= 7 THEN ARROW = 3
   IF ARROW <= 2 THEN ARROW = 6
   IF ARROW = 3 THEN WINSELECT = 1
   IF ARROW = 4 THEN WINSELECT = 2
   IF ARROW = 5 THEN WINSELECT = 3
   LOCATE OLDARROW, 3
   PRINT CHR$(251)
   LOCATE ARROW, 3
   PRINT "*"
   IF WINDOWKEY1 = 13 THEN
   WINSELECT2 = WINSELECT
IF WINSELECT2 = 1 THEN
PUT ((A - STARTJ) * 32, (B - STARTI - 1) * 32), TAKARAG%, PSET: TAKARA1 = 1
END IF
```

```
IF WINSELECT2 = 2 THEN
PUT ((A - STARTJ) * 32, (B - STARTI - 1) * 32), TAKARA0%, PSET: TAKARA1 = 1
LINE (200, 40)-(470, 110), 0, BF
            LINE (200, 40)-(470, 110), 7, B
            LINE (201, 41)-(469, 109), 7, B
            LOCATE 5, 30
            PRINT "ﾃﾞﾗｲｽ ﾊ 1000 ｺﾞｰﾙﾄﾞｦ ﾃﾆｲﾚﾀ｡ "
        END IF
    IF WINSELECT2 = 3 THEN EXIT DO
END IF
LOOP
CLS 2
PUT (10, 20), TEMPO%, PSET
PUT (200, 40), TEMPO2%, PSET
END IF
RETURN
```

2行目, IF_A_=_10_AND_B_=_10_THEN　で,『デライス』の居場所をチェックしています．これが，宝箱の隣に来たときに，それ以後の部分を実行します．

次に，このサブルーチンでは，画面の上に小さなウィンドウを開きます．ウィンドウというのは，画面の一角に，いままでの絵をどけてなんらかの情報を表示するものです．通常は，ただ情報を表示するだけでなく，矢印キーなどで，なんらかの選択をすることができ，リターンキーなどで選択した行動を行うとともに，ウィンドウそのものが消えて元の画面にもどるものです．

では，秘打ウィンドウのテクニックを紹介します．

画面に四角く黒いウィンドウを表示するのは簡単ですが，問題は元にもどるときに，最初と同じようにもどすことです．

★カラー口絵 p. 3およびカバー参照

『デライス対ゲドパ』
（ロールプレイングゲーム）
（画面にウィンドウを開いているところ）

そのためには，まず，DIM_TEMPO%(2400)　と配列を本体で宣言しておき，サブルーチンの 4 行目でウィンドウを表示する直前に，　GET_(10,_20)−(80,150),_TEMP0%　と，ウィンドウを表示する予定の場所の絵を全部取り込んでしまいます．その後で，LINE_(10,_20)−(80,_150),_0,_BF　とウィンドウのところを黒の四角で塗りつぶしてしまいます．その後，LOCATE 命令と PRINT 命令を使って，必要な情報を表示させます．

ウィンドウを消したくなった場合，CLS 2 で不要な文字を消し，最後に PUT_(10,_20),_TEMPO%,_PSET　として，前に取り込んでおいた絵を，同じように表示してウィンドウを消しています．

次に，ウィンドウの中のアケル，トル，ツカウ，オワリなどのどれを選ぶかですが，ARROW という変数を使っています．前に出てきたキー入力のルーチンを使って，下向きの矢印を押すと ARROW の値が 1 つ増えるようにします．ただし，6 より大きくなると 3 にもどるように IF_ARROW_>=_7_THEN_ARROW_=_3　の行を加えています．上向きの矢印を押すと，ARPOW の値が 1 つずつ減るようにしています．これも，2 以下になると 6 にもどるように，IF_ARROW_<=_2_THEN_ARROW_=_6　の行を加えています．また，このあとで，LOCATE_ARROW,_3:PRINT_"*"　の命令で，現在どの命令をスタンバイしているかを＊で示させています．

このままでは，前に描いた＊が消えないので，元のARROWの位置をOLDARROW として，そこに PRINT_CHR$(251)　としています．この CHR$(251) というのは，通常キャラクタに付けられた各番号には対応する文字があるのですが，この番号に対応する文字はなく，この文字をプリントすることで前の字を消すことができます．

次に，ARROW の値に応じてウィンドウを使ってやらせようとする仕事を分けています．

まず，IF_ARROW_=_3_THEN_WINSELECT_=_1　のようにやるべき仕事 WINSELECT の値を変えます．

次に，IF_WINDOWKEY1_=_13　つまり，リターンキーを押した時に初めて WINSELECT_2 に WINSELECT の値が読み込まれ，その値により次の行動を行うようになっています．

あとは，宝箱を開けるという命令に＊を合わせてリターンキーを押すと，宝箱が開いて中に金貨がある絵を表示し，トルという命令に＊を合わせてリターンを押すと，宝箱は開いてはいるが金貨がなくなった絵を表示して合わせて“ゴ

ールド”が増えたという文字を表示させ，変数GOLDの値を増加させています．

最後の方で，TAKARA1_=_1　として，次に同じ場所に来ても，もうTAKARACHECKのルーチンに行かないようにしています．これにより同じ場所に行くと，いつも同じ宝が出てくるのを防いでいます．

最初のTAKARACHECKに行くかどうかを決めるルーチンのところを，

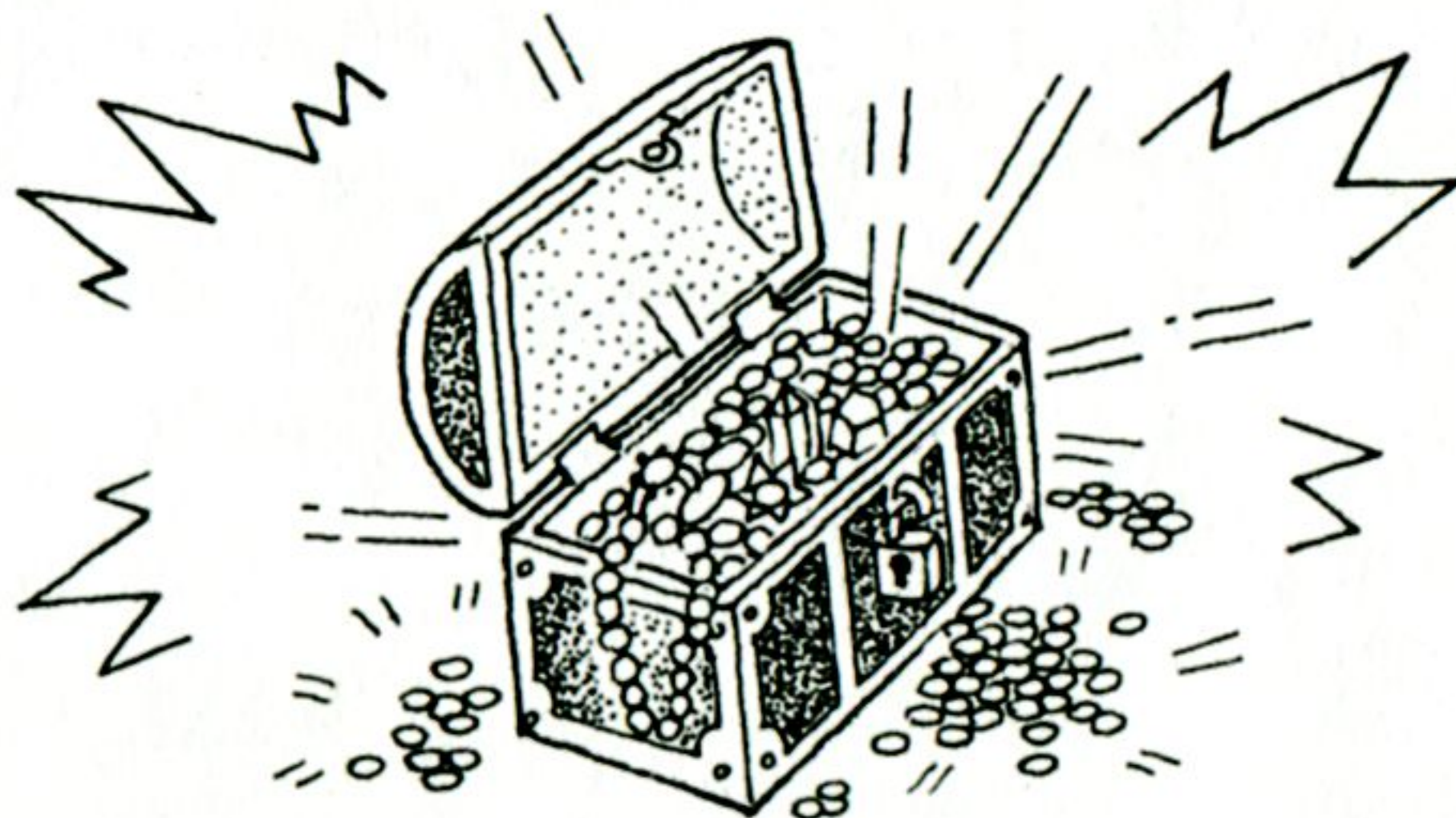
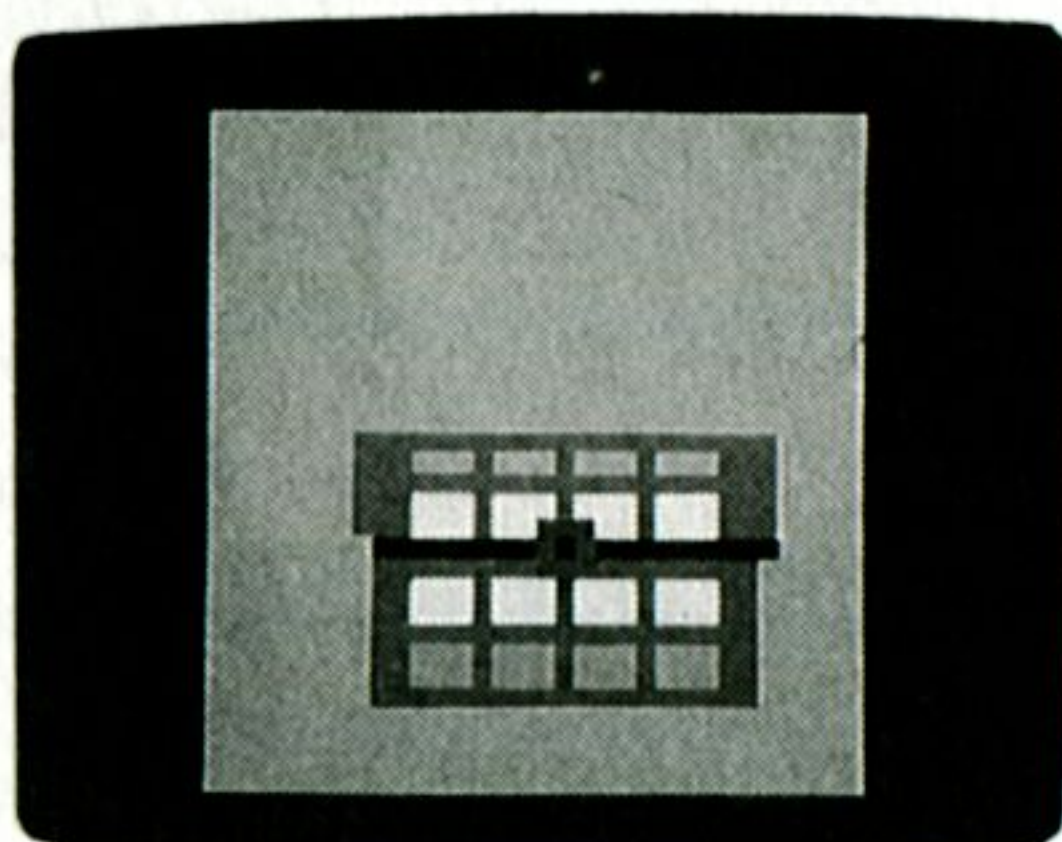
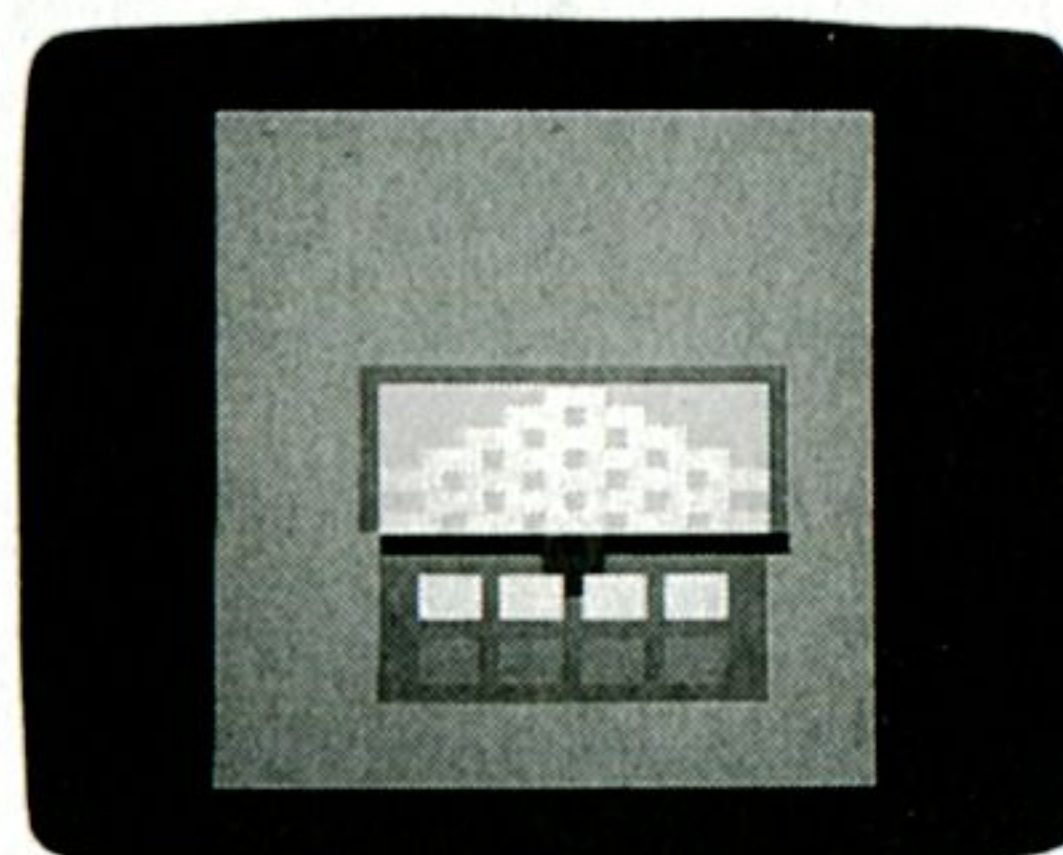
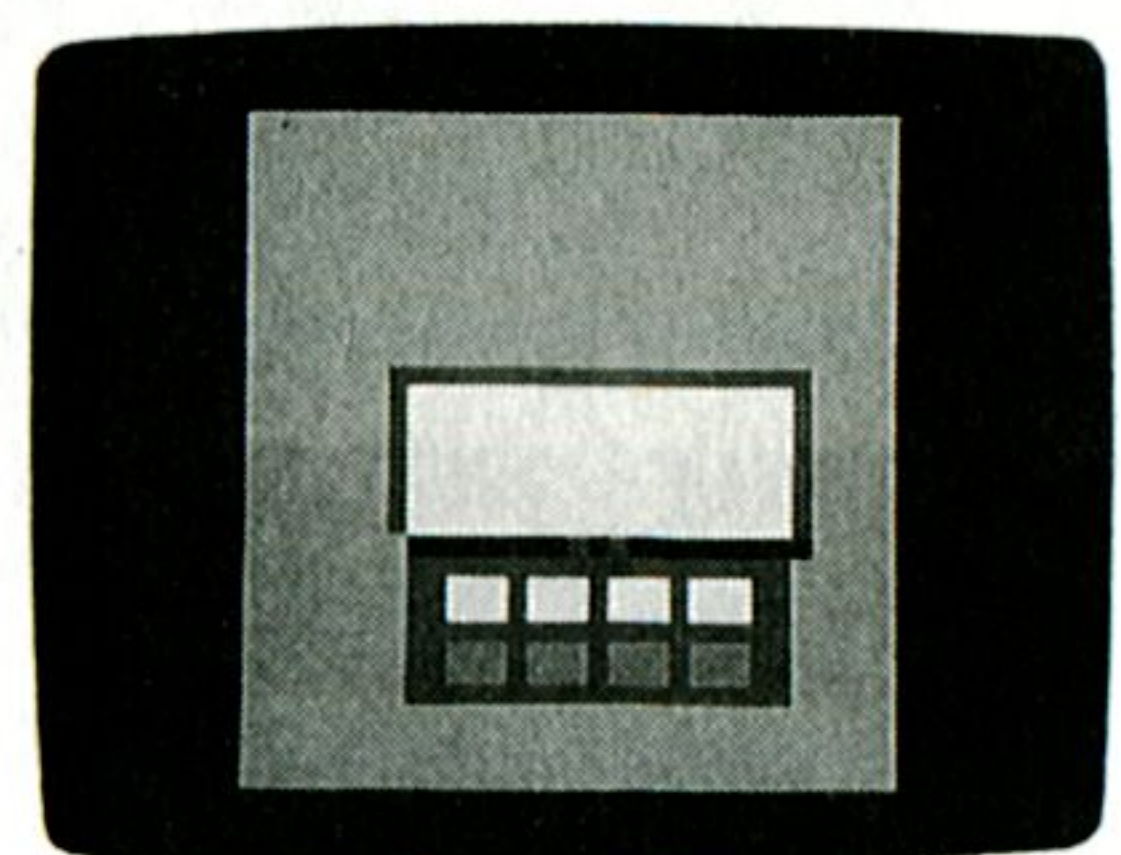
IF_TAKARA1_<_101_AND_TOWN_=_0

としておき，TAKARA1CHECKの最後の方の，

TAKARA1_=_1　を，

TAKARA1_=_TAKARA1_+_1

のように変えると，同じ所に行くと100回宝箱が出てきて，101回目からは出てきません．

このやり方は，『スーパーマリオ』のように同じところをたたくと，金貨が何枚も出てくるが，規定の枚数出ると，あとは出てこないといったときに利用できます．

★カラー口絵p.3参照

TAKARAC
（宝箱が閉まっている）

TAKARAG
（宝箱が開いて金貨が入っいる）

TAKRAO
（宝箱がカラッポ）

★「イチ，ニー，イチ，ニー」行進始め

『ドラゴンクエスト』などでは勇者が，右足を上げたり左足を上げたり歩いている状態を示しています．移動していないときにも足踏みをしているのはご愛きょうというところですが，止まったまま動かないよりはずっとましでしょう．

では，この足を動かすプログラムについて説明します．

149ページの5行目以下に示す部分が，足を交互に動かすプログラムです．

KOSHIN．BASプログラム

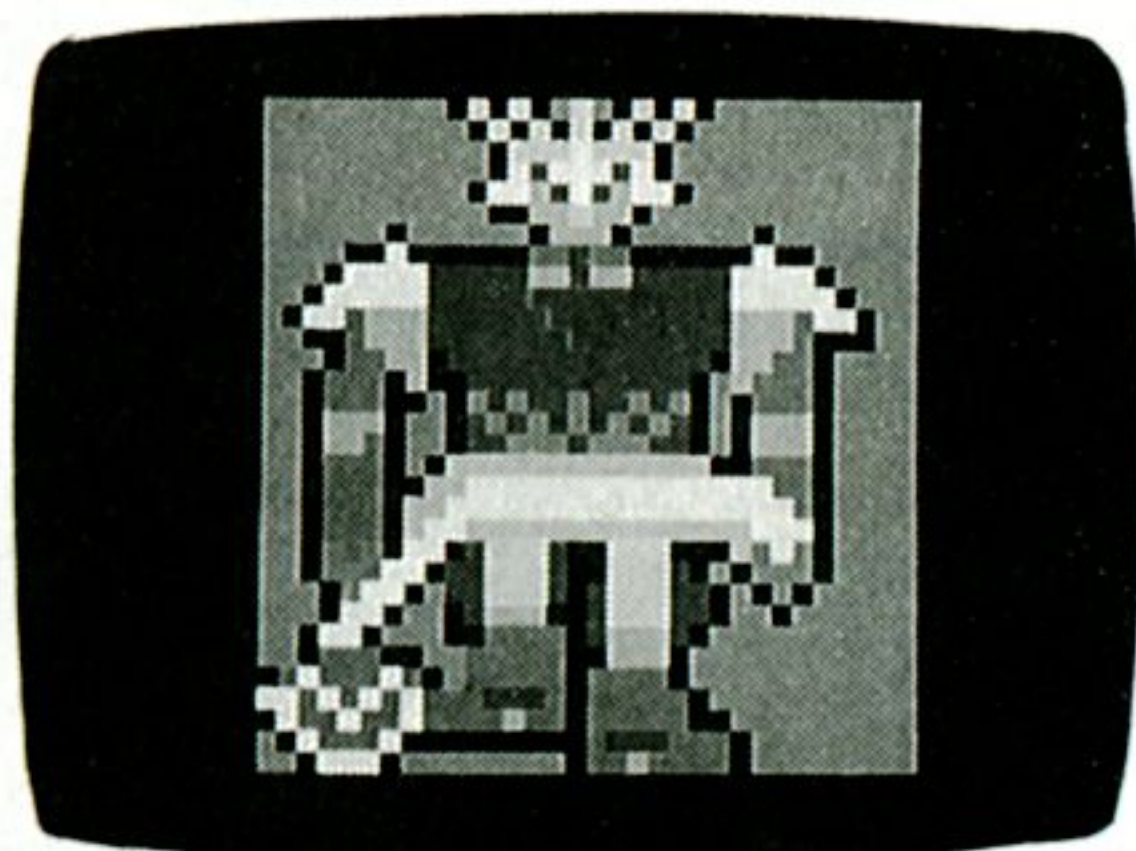
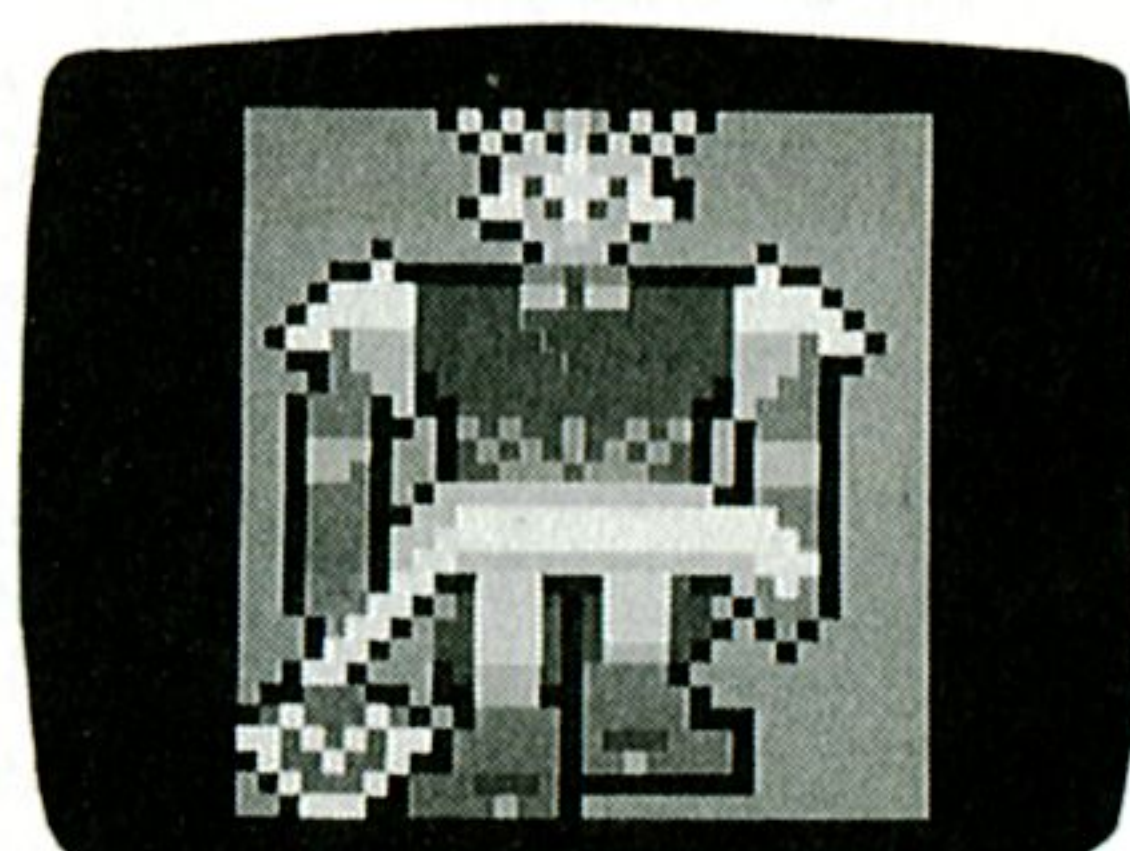
```
KEYCHECK:  C$ = INKEY$
IF TIMECK > 60 THEN TIMECK = 0
IF TIMECK > 30 THEN PUT (E * 32, F * 32), EXDESR%, PSET
IF TIMECK <= 30 THEN PUT (E * 32, F * 32), EXDESL%, PSET
TIMECK = TIMECK + 1
IF C$ = "" THEN GOTO KEYCHECK
```

このプログラムは通常，何のキー入力もない状態では一番最後の行に来ると最初のKEYCHECK: の部分にもどりこの間をずっと繰り返しています．そこで，TIMECKという変数を作り，TIMECK_＝_TIMECK_＋_1　として1回りこのループの中を通過するたびに，その値を1つずつ増やしていきます．

そのままでは値が無限に大きくなってしまいますので，IF_TIMECK_＞_60_THEN_TIMECK_＝_0　として，60を超えたら強制的に値を0にもどしています．

こうやって，変数TIMECKの値をどんどん増加させていくのですが，そのときに，IF_TIMECK_＜＝_30_THEN_PUT_(E_＊_32,_F_＊_32),_EXDESL%,_PSET として TIMECK が30以下のときには，左足が下におりている絵EXDESL%を表示し，TIMECKの値が31から60までのときには　IF_TIMECK_＞_30_THEN_PUT (E_＊_32,_F_＊_32),_EXDESR%,_PSET　右足が下におりている絵EXDESR%を表示させています．このようにして，あたかも足を交互に動かしているように見せています．足を動かす速度は上記の数字を替えて調節してください．

注）EXDESL と EXDESR の絵は，場所によって GREENSL, GREENSR または PINKSL, PINKSR を読み込んで使っているので，とくにキャラクタジェネレータで作る必要はありません．

★カラー口絵 p. 2 参照

ＤＬＶＳＧＤ．ＢＡＳプログラム

```
SCREEN 88, 3, 1, 1
CLS
WINDOW SCREEN (0, 0)-(639, 399)
DIM TEMPO%(2400)
DIM TEMPO2%(4900)
DIM TEMPO3%(4900)
ARROW = 3: OLDARROW = 3
DIM SHOUKYO%(800)
GET (0, 0)-(31, 31), SHOUKYO%
STARTJ = 0: STARTI = 2
A = STARTJ + 10: B = STARTI + 6
OLDA = A: OLDB = B
E = (A - STARTJ - 1): F = (B - STARTI - 1)
PUT (C * 32, D * 32), SHOUKYO%, PSET
C = E: D = F
TATEMAX = 29
YOKOMAX = 35
TIMECK = 0
FLOORCOLOR = 2
DIM SOTO(TATEMAX, YOKOMAX) AS STRING
DIM MAZE(TATEMAX, YOKOMAX) AS STRING
FOR I = 0 TO TATEMAX - 1
FOR J = 0 TO YOKOMAX - 1
        READ MX$
        SOTO$(I, J) = MX$
NEXT
NEXT
FOR I = 0 TO TATEMAX - 1
FOR J = 0 TO YOKOMAX - 1
        MAZE$(I, J) = SOTO(I, J)
NEXT
NEXT
RESTORE 20000
DIM MACHIA(TATEMAX, YOKOMAX) AS STRING
FOR I = 0 TO TATEMAX - 1
FOR J = 0 TO YOKOMAX - 1
        READ MX$
        MACHIA$(I, J) = MX$
```

```
NEXT
NEXT
DIM GRASS%(800)
    DEF SEG = VARSEG(GRASS%(0))
    OFFSET% = VARPTR(GRASS%(0))
    BLOAD "GRASS.GRA", OFFSET%
    DEF SEG
DIM GREEN%(800)
    DEF SEG = VARSEG(GREEN%(0))
    OFFSET% = VARPTR(GREEN%(0))
    BLOAD "GREEN.GRA", OFFSET%
    DEF SEG
DIM PINK%(800)
    DEF SEG = VARSEG(PINK%(0))
    OFFSET% = VARPTR(PINK%(0))
    BLOAD "PINK.GRA", OFFSET%
    DEF SEG
DIM TREE%(800)
    DEF SEG = VARSEG(TREE%(0))
    OFFSET% = VARPTR(TREE%(0))
    BLOAD "TREE.GRA", OFFSET%
    DEF SEG
DIM WATER%(800)
    DEF SEG = VARSEG(WATER%(0))
    OFFSET% = VARPTR(WATER%(0))
    BLOAD "WATER.GRA", OFFSET%
    DEF SEG
DIM CASTLLB%(800)
    DEF SEG = VARSEG(CASTLLB%(0))
    OFFSET% = VARPTR(CASTLLB%(0))
    BLOAD "CASTLLB.GRA", OFFSET%
    DEF SEG
DIM CASTLRT%(800)
    DEF SEG = VARSEG(CASTLRT%(0))
    OFFSET% = VARPTR(CASTLRT%(0))
    BLOAD "CASTLRT.GRA", OFFSET%
    DEF SEG
DIM CASTLLT%(800)
    DEF SEG = VARSEG(CASTLLT%(0))
    OFFSET% = VARPTR(CASTLLT%(0))
    BLOAD "CASTLLT.GRA", OFFSET%
    DEF SEG
DIM CASTLRB%(800)
    DEF SEG = VARSEG(CASTLRB%(0))
    OFFSET% = VARPTR(CASTLRB%(0))
    BLOAD "CASTLRB.GRA", OFFSET%
    DEF SEG
DIM EXDESR%(800)
DIM EXDESL%(800)
DIM GREENSS%(800)
    NAME$ = "GREENSS"
    DEF SEG = VARSEG(GREENSS%(0))
    OFFSET2% = VARPTR(GREENSS%(0))
```

```
      BLOAD NAME$ + ".GRA", OFFSET2%
  DIM GREENSR%(800)
      NAME$ = "GREENSR"
      DEF SEG = VARSEG(GREENSR%(0))
      OFFSET2% = VARPTR(GREENSR%(0))
      BLOAD NAME$ + ".GRA", OFFSET2%
      FOR I = 0 TO 800
      EXDESR%(I) = GREENSR%(I)
      NEXT I
  DIM GREENSL%(800)
      NAME$ = "GREENSL"
      DEF SEG = VARSEG(GREENSL%(0))
      OFFSET2% = VARPTR(GREENSL%(0))
      BLOAD NAME$ + ".GRA", OFFSET2%
      FOR I = 0 TO 800
      EXDESL%(I) = GREENSL%(I)
      NEXT I
  DIM PINKSS%(800)
      NAME$ = "PINKSS"
      DEF SEG = VARSEG(PINKSS%(0))
      OFFSET2% = VARPTR(PINKSS%(0))
      BLOAD NAME$ + ".GRA", OFFSET2%
  DIM PINKSR%(800)
      NAME$ = "PINKSR"
      DEF SEG = VARSEG(PINKSR%(0))
      OFFSET2% = VARPTR(PINKSR%(0))
      BLOAD NAME$ + ".GRA", OFFSET2%
  DIM PINKSL%(800)
      NAME$ = "PINKSL"
      DEF SEG = VARSEG(PINKSL%(0))
      OFFSET2% = VARPTR(PINKSL%(0))
      BLOAD NAME$ + ".GRA", OFFSET2%
  DIM TAKARAC%(800)
      DEF SEG = VARSEG(TAKARAC%(0))
      OFFSET% = VARPTR(TAKARAC%(0))
      BLOAD "TAKARAC.GRA", OFFSET%
      DEF SEG
  DIM TAKARAG%(800)
      DEF SEG = VARSEG(TAKARAG%(0))
      OFFSET% = VARPTR(TAKARAG%(0))
      BLOAD "TAKARAG.GRA", OFFSET%
      DEF SEG
  DIM TAKARAO%(800)
      DEF SEG = VARSEG(TAKARAO%(0))
      OFFSET% = VARPTR(TAKARAO%(0))
      BLOAD "TAKARAO.GRA", OFFSET%
      DEF SEG
  DIM BRICK%(800)
      DEF SEG = VARSEG(BRICK%(0))
      OFFSET% = VARPTR(BRICK%(0))
      BLOAD "BRICK.GRA", OFFSET%
      DEF SEG
  DIM KING%(800)
```

```
        DEF SEG = VARSEG(KING%(0))
        OFFSET% = VARPTR(KING%(0))
        BLOAD "KING.GRA", OFFSET%
        DEF SEG
  START:
  CLS
  PAINT (200, 200), FLOORCOLOR
  IF STARTJ > YOKOMAX - 22 THEN STARTJ = YOKOMAX - 22
  IF STARTJ < 4 THEN STARTJ = 0
  IF STARTI > TATEMAX - 14 THEN STARTI = TATEMAX - 14
  IF STARTI < 4 THEN STARTI = 0
  FOR I = STARTI + 1 TO STARTI + 12
  FOR J = STARTJ + 1 TO STARTJ + 20
          FX$ = MAZE$(I, J)
  IF FX$ = "1" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), GRASS%, PSET
  IF FX$ = "4" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), WATER%, PSET
  IF FX$ = "5" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), TREE%, PSET
  IF FX$ = "6" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLLB%, PS
ET
  IF FX$ = "7" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLLT%, PS
ET
  IF FX$ = "8" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLRT%, PS
ET
  IF FX$ = "9" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), CASTLRB%, PS
ET
  IF FX$ = "A" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), BRICK%, PSET
  IF FX$ = "K" THEN PUT ((J - STARTJ - 1) * 32, (I - STARTI - 1) * 32), KING%, PSET
  NEXT
  NEXT
  E = (A - STARTJ - 1): F = (B - STARTI - 1)
  C = E: D = F
  REM PUT (C * 32, D * 32), SHOUKYO%, PSET
  OLDFX$ = MAZE$(OLDB, OLDA)
  IF OLDFX$ = "0" AND TOWN = 0 THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI -
1) * 32), GREEN%, PSET
  IF OLDFX$ = "0" AND TOWN = 1 THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI -
1) * 32), PINK%, PSET
  IF OLDFX$ = "1" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), GRA
SS%, PSET
  IF OLDFX$ = "4" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), WAT
ER%, PSET
  IF OLDFX$ = "5" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), TRE
E%, PSET
  IF OLDFX$ = "6" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLLB%, PSET
  IF OLDFX$ = "7" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLLT%, PSET
  IF OLDFX$ = "8" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLRT%, PSET
  IF OLDFX$ = "9" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLRB%, PSET
  IF OLDFX$ = "A" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), BRI
CK%, PSET
```

```
  IF OLDFX$ = "K" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), KIN
G%, PSET
  IF TOWN = 0 THEN PUT (E * 32, F * 32), GREENSS%, PSET
  IF TOWN = 1 THEN PUT (E * 32, F * 32), PINKSS%, PSET
  KEYCHECK:  C$ = INKEY$
  IF TIMECK > 60 THEN TIMECK = 0
  IF TIMECK > 30 THEN PUT (E * 32, F * 32), EXDESR%, PSET
  IF TIMECK <= 30 THEN PUT (E * 32, F * 32), EXDESL%, PSET
  TIMECK = TIMECK + 1
  IF C$ = "" THEN GOTO KEYCHECK
      IF C$ = "8" AND MAZE(B - 1, A) < "4" THEN
      A = A: B = B - 1: OLDA = A: OLDB = B + 1: GOSUB MOVE
      END IF
      IF C$ = "2" AND MAZE(B + 1, A) < "4" THEN
      A = A: B = B + 1: OLDA = A: OLDB = B - 1: GOSUB MOVE
      END IF
      IF C$ = "4" AND MAZE(B, A - 1) < "4" THEN
      A = A - 1: B = B: OLDA = A + 1: OLDB = B: GOSUB MOVE
      END IF
      IF C$ = "6" AND MAZE(B, A + 1) < "4" THEN
      A = A + 1: B = B: OLDA = A - 1: OLDB = B: GOSUB MOVE
      END IF
      IF C$ = "7" AND MAZE(B - 1, A - 1) < "4" THEN
      A = A - 1: B = B - 1: OLDA = A + 1: OLDB = B + 1: GOSUB MOVE
      END IF
      IF C$ = "9" AND MAZE(B - 1, A + 1) < "4" THEN
      A = A + 1: B = B - 1: OLDA = A - 1: OLDB = B + 1: GOSUB MOVE
      END IF
      IF C$ = "1" AND MAZE(B + 1, A - 1) < "4" THEN
      A = A - 1: B = B + 1: OLDA = A + 1: OLDB = B - 1: GOSUB MOVE
      END IF
      IF C$ = "3" AND MAZE(B + 1, A + 1) < "4" THEN
      A = A + 1: B = B + 1: OLDA = A - 1: OLDB = B - 1: GOSUB MOVE
      END IF
      IF C$ = "Q" THEN END
      IF C$ = "q" THEN END
 KEY1 = ASC(LEFT$(C$, 1))
 KEY2 = ASC(RIGHT$(C$, 1))
    IF KEY1 = 0 AND KEY2 = 75 AND MAZE(B, A - 1) < "4" THEN
    A = A - 1: B = B: OLDA = A + 1: OLDB = B: GOSUB MOVE
    END IF
    IF KEY1 = 0 AND KEY2 = 77 AND MAZE(B, A + 1) < "4" THEN
    A = A + 1: B = B: OLDA = A - 1: OLDB = B: GOSUB MOVE
    END IF
    IF KEY1 = 0 AND KEY2 = 72 AND MAZE(B - 1, A) < "4" THEN
    A = A: B = B - 1: OLDA = A: OLDB = B + 1: GOSUB MOVE
    END IF
    IF KEY1 = 0 AND KEY2 = 80 AND MAZE(B + 1, A) < "4" THEN
    A = A: B = B + 1: OLDA = A: OLDB = B - 1: GOSUB MOVE
    END IF
 IF B >= 4 AND B <= 7 AND A >= 12 AND A <= 15 AND TOWN = 0 THEN
 FOR I = 0 TO TATEMAX - 1
 FOR J = 0 TO YOKOMAX - 1
```

```
        MAZE$(I, J) = MACHIA$(I, J)
 NEXT
 NEXT
 TOWN = 1
 STARTI = 22
 STARTJ = 5
 OLDA = STARTJ + 1
 OLDB = STARTI + 1
 A = 16
 B = 26
 FOR I = 0 TO 800
 EXDESR%(I) = PINKSR%(I)
 NEXT I
 FOR I = 0 TO 800
 EXDESL%(I) = PINKSL%(I)
 NEXT I
 FLOORCOLOR = 5
 GOTO START
 END IF
 IF B >= 26 AND A >= 12 AND A <= 20 AND TOWN = 1 THEN
 FOR I = 0 TO TATEMAX - 1
 FOR J = 0 TO YOKOMAX - 1
        MAZE$(I, J) = SOTO$(I, J)
 NEXT
 NEXT
 TOWN = 0
 STARTI = 2
 STARTJ = 5
 OLDA = STARTJ + 1
 OLDB = STARTI + 1
 A = 13
 B = 8
 FOR I = 0 TO 800
 EXDESR%(I) = GREENSR%(I)
 NEXT I
 FOR I = 0 TO 800
 EXDESL%(I) = GREENSL%(I)
 NEXT I
 FLOORCOLOR = 2
 GOTO START
 END IF
 IF TAKARA1 = 0 AND TOWN = 0 THEN GOSUB TAKARACHECK
 IF TOWN = 1 THEN GOSUB KINGCHECK
 GOTO KEYCHECK
 MOVE:
 IF A < 1 THEN A = 1
 IF A > YOKOMAX - 1 THEN A = YOKOMAX
 IF B < 1 THEN B = 1
 IF B > TATEMAX - 1 THEN B = TATEMAX
 OLDFX$ = MAZE$(OLDB, OLDA)
 IF OLDFX$ = "0" AND TOWN = 0 THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI -
1) * 32), GREEN%, PSET
 IF OLDFX$ = "0" AND TOWN = 1 THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI -
```

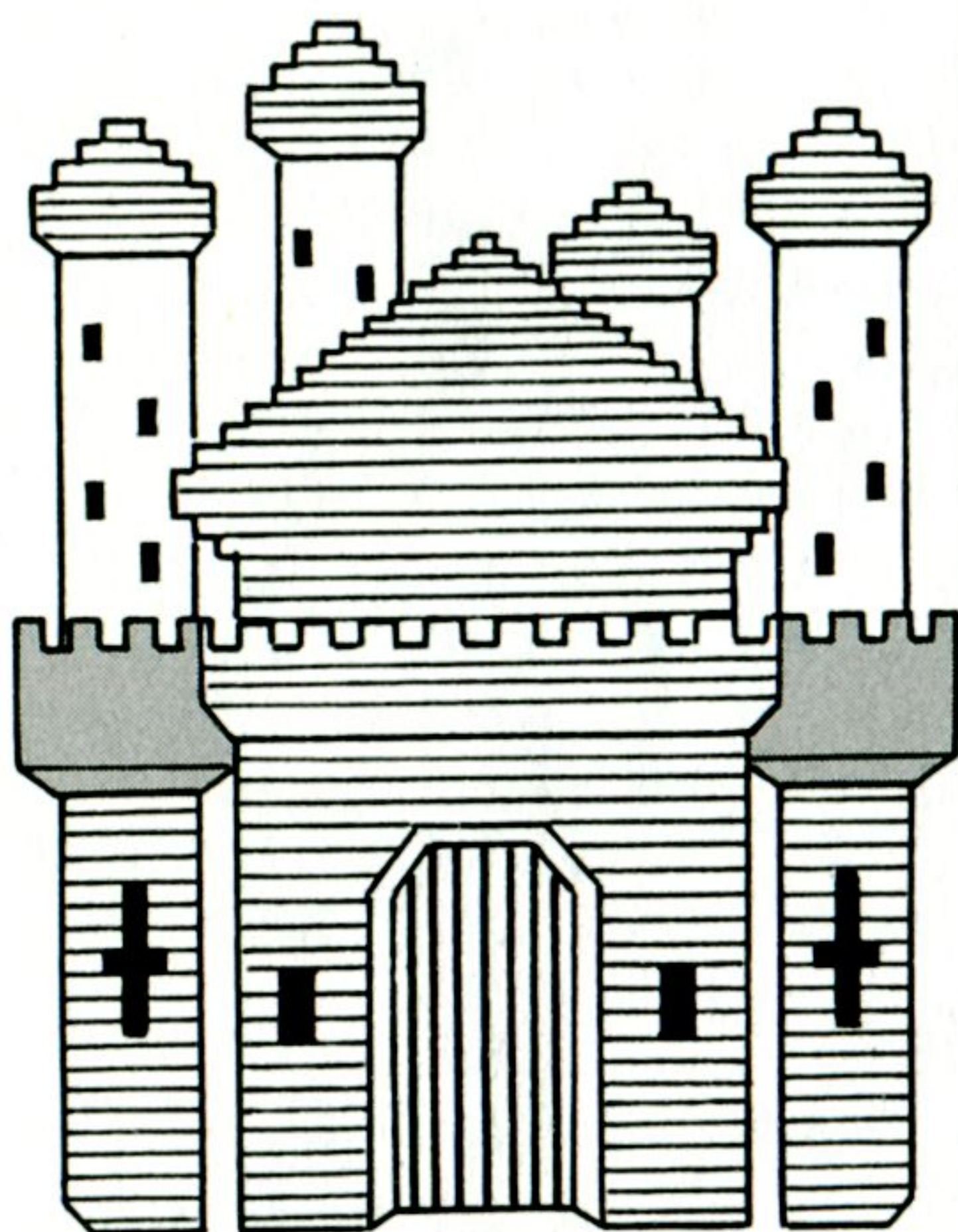
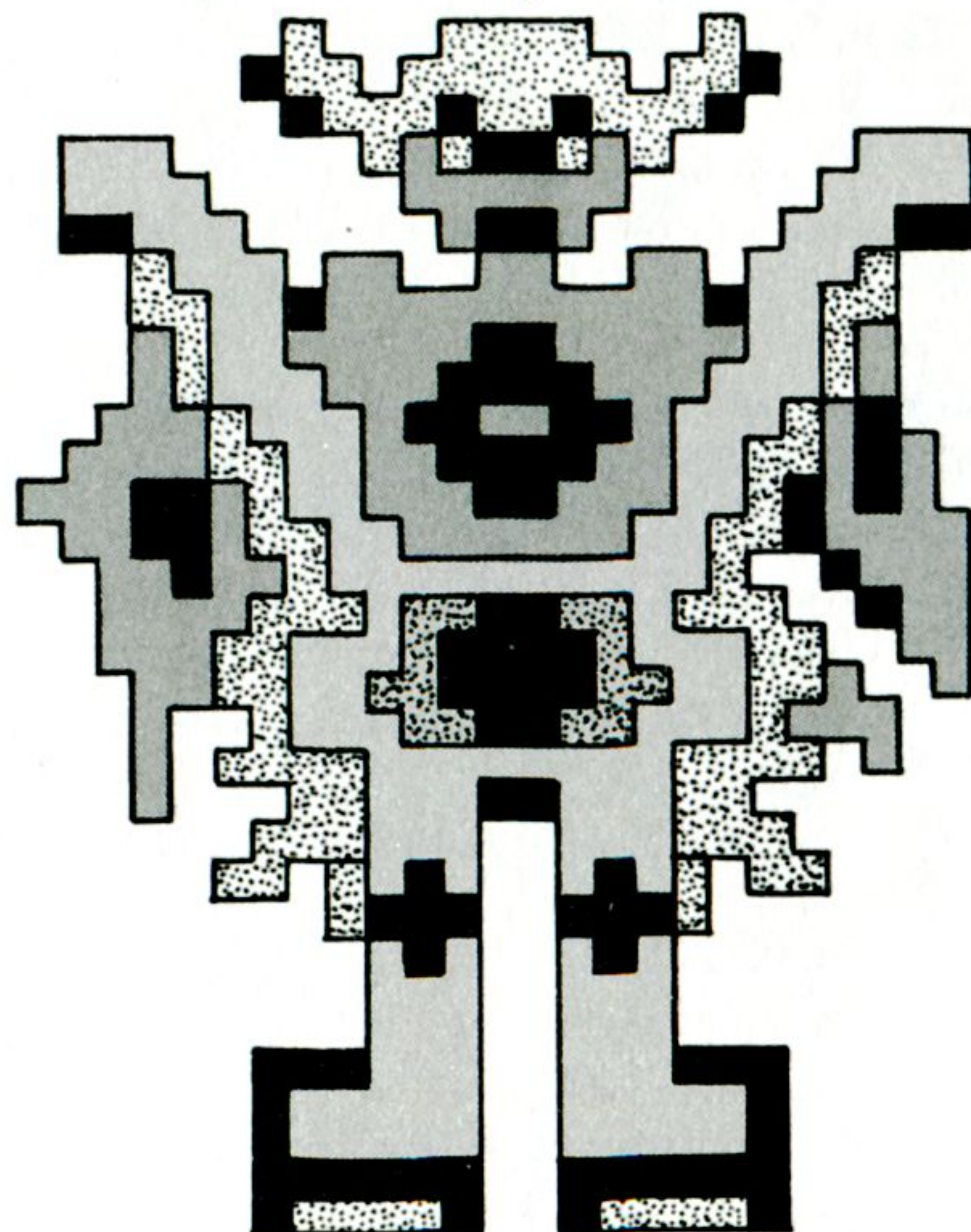
```
1) * 32), PINK%, PSET
  IF OLDFX$ = "1" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), GRA
SS%, PSET
  IF OLDFX$ = "4" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), WAT
ER%, PSET
  IF OLDFX$ = "5" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), TRE
E%, PSET
  IF OLDFX$ = "6" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLLB%, PSET
  IF OLDFX$ = "7" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLLT%, PSET
  IF OLDFX$ = "8" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLRT%, PSET
  IF OLDFX$ = "9" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), CAS
TLRB%, PSET
  IF OLDFX$ = "A" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), BRI
CK%, PSET
  IF OLDFX$ = "K" THEN PUT ((OLDA - STARTJ - 1) * 32, (OLDB - STARTI - 1) * 32), KIN
G%, PSET
  E = (A - STARTJ - 1): F = (B - STARTI - 1)
  IF E > 16 AND (YOKOMAX - A) > 7 THEN STARTJ = STARTJ + 7: GOTO START
  IF E < 4 AND A > 7 THEN STARTJ = STARTJ - 5: GOTO START
  IF F > 10 AND (TATEMAX - B) > 4 THEN STARTI = STARTI + 4: GOTO START
  IF F < 2 AND B > 4 THEN STARTI = STARTI - 4: GOTO START
  IF E <= 0 THEN E = 0
  IF E >= 19 THEN E = 19
  IF F <= 0 THEN F = 0
  IF F >= 11 THEN F = 11
  IF TOWN = 0 THEN PUT (E * 32, F * 32), GREENSS%, PSET
  IF TOWN = 1 THEN PUT (E * 32, F * 32), PINKSS%, PSET
  C = E: D = F
  RETURN
  TAKARACHECK:
  IF A = 10 AND B = 10 THEN
  PUT ((A - STARTJ) * 32, (B - STARTI - 1) * 32), TAKARAC%, PSET
  GET (10, 20)-(80, 150), TEMPO%
  GET (200, 40)-(470, 110), TEMPO2%
  LINE (10, 20)-(80, 150), 0, BF
  LINE (10, 20)-(80, 150), 7, BF
  LINE (11, 21)-(79, 149), 0, BF
  CLS 2
  LOCATE 3, 5: PRINT "OPEN"
  LOCATE 4, 5: PRINT "TAKE"
  LOCATE 5, 5: PRINT "USE"
  LOCATE 6, 5: PRINT "END"
  DO WHILE 1
  WINDOW2:
     LOCATE ARROW, 3
     PRINT "*"
  WINDOWKEY$ = INKEY$
  IF WINDOWKEY$ = "" THEN GOTO WINDOW2
  WINDOWKEY1 = ASC(LEFT$(WINDOWKEY$, 1))
  WINDOWKEY2 = ASC(RIGHT$(WINDOWKEY$, 1))
```

```
    IF WINDOWKEY1 = 0 AND WINDOWKEY2 = 72 THEN
    ARROW = ARROW - 1: OLDARROW = ARROW + 1
    END IF
    IF WINDOWKEY1 = 0 AND WINDOWKEY2 = 80 THEN
    ARROW = ARROW + 1: OLDARROW = ARROW - 1
    END IF
    IF ARROW >= 7 THEN ARROW = 3
    IF ARROW <= 2 THEN ARROW = 6
    IF ARROW = 3 THEN WINSELECT = 1
    IF ARROW = 4 THEN WINSELECT = 2
    IF ARROW = 5 THEN WINSELECT = 3
    LOCATE OLDARROW, 3
    PRINT CHR$(251)
    LOCATE ARROW, 3
    PRINT "*"
    IF WINDOWKEY1 = 13 THEN
    WINSELECT2 = WINSELECT
IF WINSELECT2 = 1 THEN
PUT ((A - STARTJ) * 32, (B - STARTI - 1) * 32), TAKARAG%, PSET: TAKARA1 = 1
END IF
IF WINSELECT2 = 2 THEN
PUT ((A - STARTJ) * 32, (B - STARTI - 1) * 32), TAKARAO%, PSET: TAKARA1 = 1
LINE (200, 40)-(470, 110), 0, BF
             LINE (200, 40)-(470, 110), 7, B
             LINE (201, 41)-(469, 109), 7, B
             LOCATE 5, 30
              PRINT "ﾃﾞﾗｲｽ ﾊ 1000 ｺﾞｰﾙﾄﾞｦ ﾃﾆｲﾚﾀ｡ "
         END IF
     IF WINSELECT2 = 3 THEN EXIT DO
 END IF
 LOOP
 CLS 2
 PUT (10, 20), TEMPO%, PSET
 PUT (200, 40), TEMPO2%, PSET
 END IF
 RETURN
 KINGCHECK:
 IF A = 16 AND B = 3 THEN
 GET (20, 40)-(290, 110), TEMPO3%
 LINE (20, 40)-(290, 110), 7, BF
 COLOR 0
 FOR I = 0 TO 3000
 LOCATE 4, 5: PRINT "ﾃﾞﾗｲｽﾖ ﾋﾒｶﾞ ｱｸﾉ ｺﾞﾝｹﾞ ｹﾞﾄﾞﾊﾞﾆ "
 LOCATE 5, 5: PRINT "ｻﾗﾜﾚﾃ ｼﾏｯﾀ｡"
 LOCATE 6, 5: PRINT "ﾋﾒｦ ﾀｽｹﾀﾞｼﾃ ﾎｼｲ｡"
 NEXT I
 COLOR 7
 CLS 2
 PUT (20, 40), TEMPO3%, PSET
 END IF
 RETURN
```

```
DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
DATA 9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,9
DATA 9,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,5,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,0,0,0,0,0,0,0,0,0,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,0,5,0,0,0,1,0,0,0,4,0,0,0,0,4,0,0,0,5,0,0,0,0,1,0,0,0,0,0,0,0,5,9
DATA 9,5,0,5,5,5,5,0,1,0,0,4,0,7,8,0,4,0,0,0,5,0,0,0,0,1,0,0,0,0,0,0,0,5,9
DATA 9,5,0,0,0,0,0,0,0,0,0,4,0,6,9,0,4,0,0,0,5,0,0,0,0,0,0,0,0,5,0,0,5,5,9
DATA 9,5,5,5,0,5,0,0,0,0,0,4,0,0,0,0,4,0,0,0,5,0,0,0,0,0,0,0,0,5,0,0,5,5,9
DATA 9,5,0,0,0,5,0,0,0,0,0,4,4,0,0,4,4,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,5,5,9
DATA 9,5,0,5,5,5,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,9
DATA 9,0,0,5,0,0,0,5,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,9
DATA 9,0,0,0,0,5,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9
DATA 9,5,5,5,0,5,0,0,0,0,0,4,4,4,4,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,0,0,0,5,0,0,0,0,0,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,0,5,5,5,5,5,0,0,0,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,0,0,5,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,5,5,0,0,0,0,5,0,5,5,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,5,5,0,5,0,0,0,0,0,4,4,4,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,0,0,0,5,0,0,0,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,5,9
DATA 9,5,0,5,5,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,5,9
DATA 9,0,0,5,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,0,5,5,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,5,9
DATA 9,0,0,5,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,9
DATA 9,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,9
DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
20000
DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
DATA 9,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,K,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,A,A,0,0,0,A,A,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A.9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,A,A,A,0,0,0,A,A,A,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
```

```
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,A,9
DATA 9,A,A,A,A,A,A,A,A,A,A,A,A,A,1,1,1,1,1,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,9
DATA 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
```

(プログラムを完全に入力したにもかかわらず動作しない場合は，11ページの囲み，および90ページのコラムを参考にしてみてください)

Quick BASIC "We were hit!"

★トム・クルーズよ早くこい

映画「トップガン」の中で，アイスマンがミグの機関銃に撃たれるシーンがありました．感情を表さないはずの彼の声がうわずっていたのもうなずけます．機関銃で撃たれちゃ大変です．

ゲームの中でも，ミサイル・爆弾との衝突，敵との出合いなど，衝突の判定は重要です．では，衝突の判定について見てみましょう．

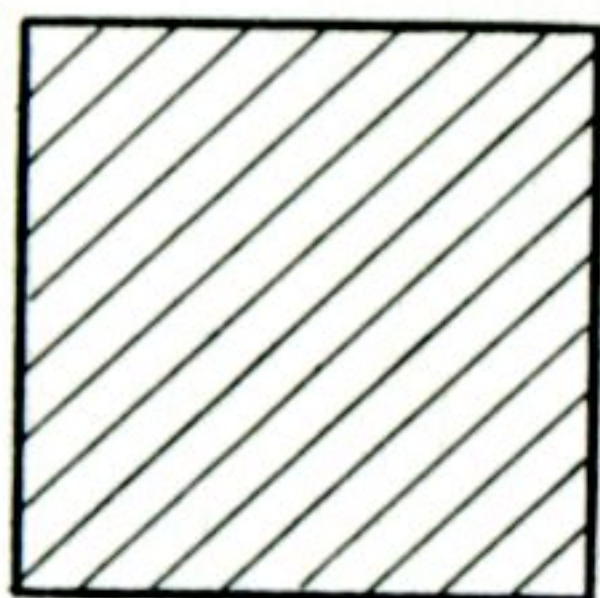
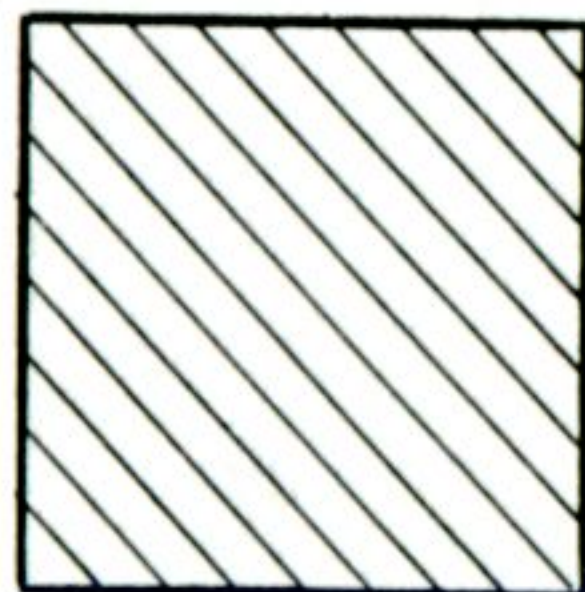
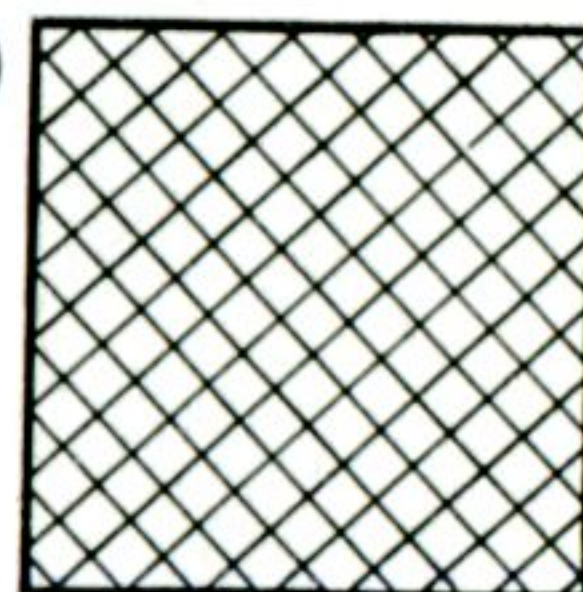
★衝突判定

自分でゲームを作ることの利点の１つは，衝突判定の基準を，自分の思いどおりにできることです．でも，あまり自分だけに都合が良すぎるようにしてしまうと，ゲームがつまらなくなってしまいます．良い審判になってください．

衝突判定の基準は，判定する２つがどれぐらい近くの場所に存在するかです．

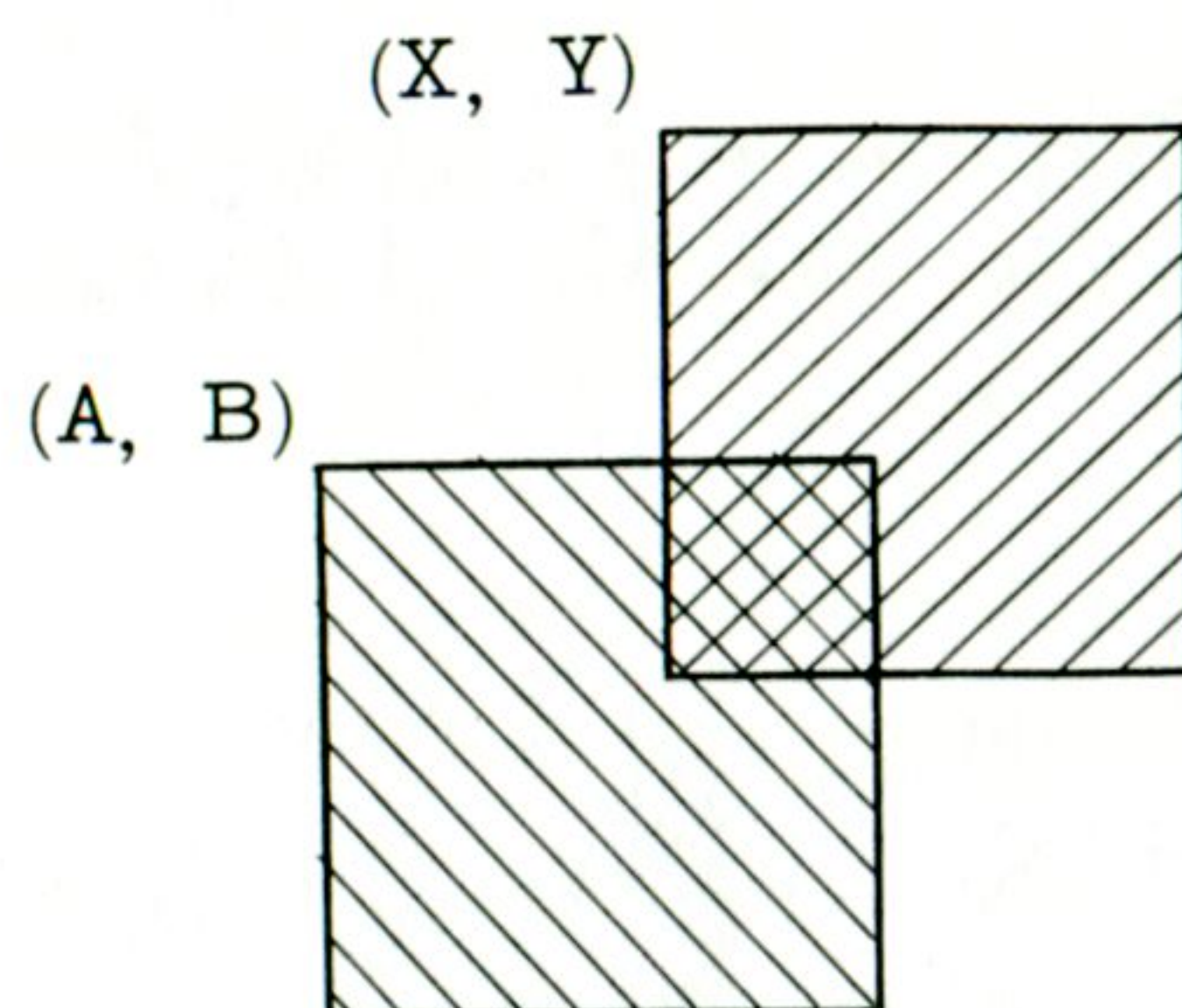
TEKI%, MIKATA%というキャラクタを，ともに32×32ドットの大きさがあるとして考えます．

いま，TEKI%というキャラクタを表示する画面上の位置をX, Yとして，MIKATA%というキャラクタを表示する位置をA, Bとすると，両者が完全に一致するときは，当然，A=X, B=Yになります．

(X, Y)　　(A, B)　　(X, Y)
(A, B)

完全衝突

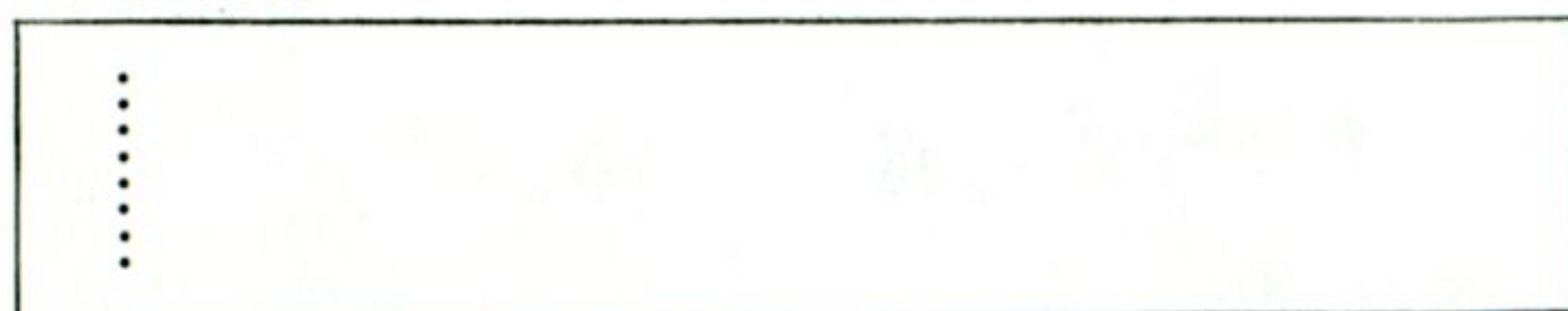
ところが普通，衝突を判定する場合には，完全に重なり合わなくても，ある程度重なったところで衝突と判定します.

(X, Y)

(A, B)

部分衝突

シミュレーション，ウォーゲームのように隣接しただけで戦いが始まるようなときの判定は，次のように行います.

衝突プログラム

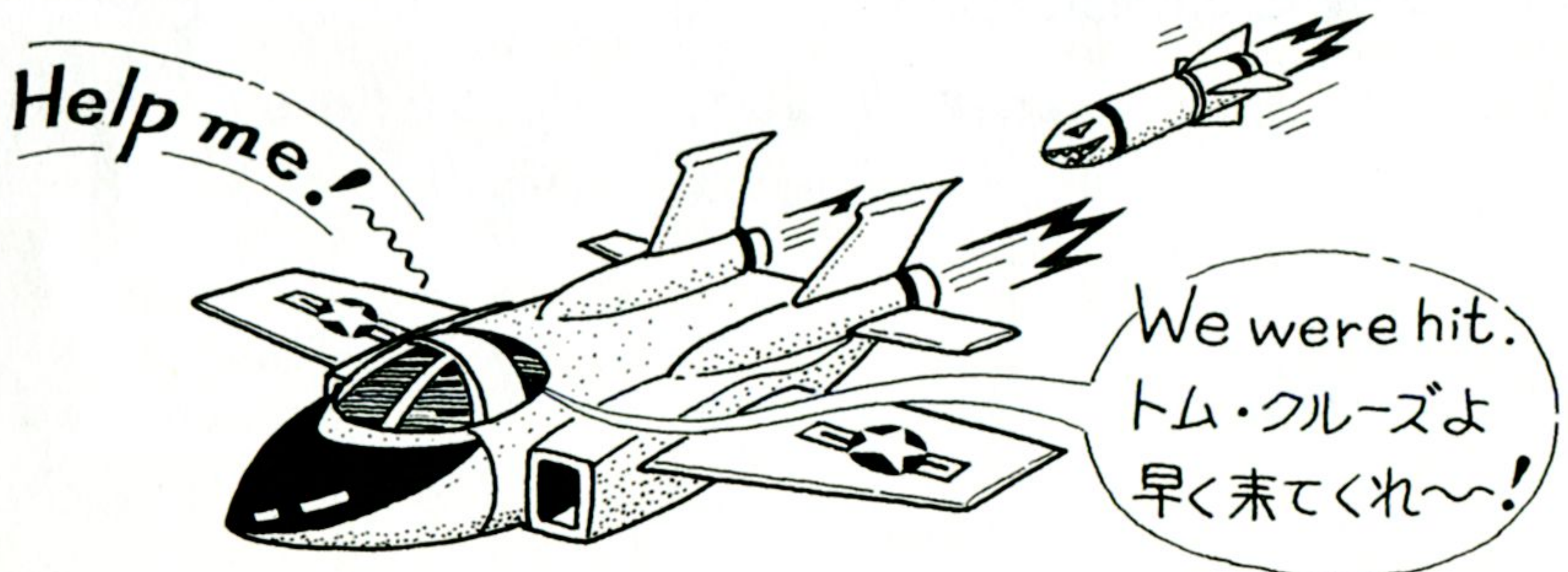
```
IF_ABS_(A_-_X)_<=_32_AND_ABS_(B_-_Y)_<=32_THEN
GOSUB_BATTLE
END_IF
    :
    :
 END

 BATTLE:
  ┌──────────────────┐
  │ :                │
  │ :                │
  └──────────────────┘
 RETURN
```

まず，TEKI%,_MIKATA% の両者を表示する横方向の位置を A−X で比べます．ABS_(A_−_X)としているのは両者の絶対値を取っているためで，これはずれ方により A−X の値がマイナスになっても正しく判定できるようにするためです．次の _＜＝_32 で両者の横方向のズレが32ドット以内になったかどうかを確かめています．縦方向の位置を表す B と Y との差についても同じように，ABS_(B_−_Y)＜＝_32 で，32ドット以内かを確かめています．そしてここで，両方の式の間にANDという言葉が出てきました．

これは，AND の前に書かれた式と，あとに書かれた式が，両方同時に成立するかどうかを見るための命令です．横方向はしっかり重なっていても，縦に離れていたら衝突とはいえません．そこで，横も縦も両方ともに条件が満たされたとき始めて THEN 以下を実施するようにするのがこの AND 命令の役目です．

成立したときには，BATTLE というサブルーチンに飛ばしています．これは，モジュールにしてそこに飛ばしても構いません．戦いのように複雑な部分はまとめておいて，そこに飛んで行かせるようにしておいた方が簡単です．

モビルスーツ同士の戦いや，敵のキャラクタと味方のキャラクタとの戦いなどでは，絵の周りに余白があるときがあります．このような場合，衝突判定の値を32以下としておくと，スレ違っただけで戦いが始まってしまいます．それがいやな人は，32という数字を28とか24とか自分の望む値に小さくしてください．

★いつ衝突判定をするか？

衝突判定は，敵の移動後，味方の移動後，それぞれについて行った方が良いでしょう．2回やるのは面倒かも知れませんが，まとめてやっていると，味方が移動して重なり合ったが，その途端敵が移動して衝突が起こらないなどと変な形になってしまいます．

★ゴースト登場(衝突判定の中のオバケの世界)

ゴーストといっても,“ロールプレイイングゲーム”に出てくるオバケのことではありません.

衝突判定は,相手を画面のどこに表示するかを示すX,Yなどの位置を判定の基準に使っています.ここで注意しなければいけないのは,相手をやっつけたあともこのXとYの値が残ることです.これを上手(うま)く処理しないと,何もないところで,敵の爆発シーンだけが出てきたりします.これを防ぐには,次のようにします.

TEKIAが生きているかどうかを示す変数TEKILIFEAを作り,これにフラッグを立てます.つまり,TEKIAが生きているときには TEKILIFEA_=_1,死んでしまったときは TEKILIFEA_=_0 とします.

画面に表示する位置をX,_Yとすると,

```
IF_TEKILIFEA_=_1_THEN_PUT_TEKIA%(X,_Y)
```

のように,画面に表示する前に必ずTEKILIFEAが1かどうかを確認します.

また,X,_Yの位置を計算して,画面に表示する部分をサブルーチン化してTEKILIFEAが1のときだけそのルーチンに飛ぶようにした方がムダな計算がないだけ優れています.

同じように衝突判定の部分もサブルーチン化して,TEKILIFEAが1のときだけTEKIAについて衝突判定をしてやる必要があります.

これと同時に,BATTLEとかTEKIBAKUHATSUのルーチーンの中でTEKIAが死んだときには, TEKILIFEA_=_0 としてTEKILIFEAの値を0にしておいてください.これで,画面をゴーストが徘徊(はいかい)することはなくなるはずです.

★複数の敵との衝突判定

敵が複数出てきたとき,敵の数だけ衝突判定のルーチンを作ると,プログラムが長大化してしまいます.そこで,同じサブルーチンをうまく使ってやりましょう.

Quick BASIC

本格的スクロールゲーム『SPACEWAR』

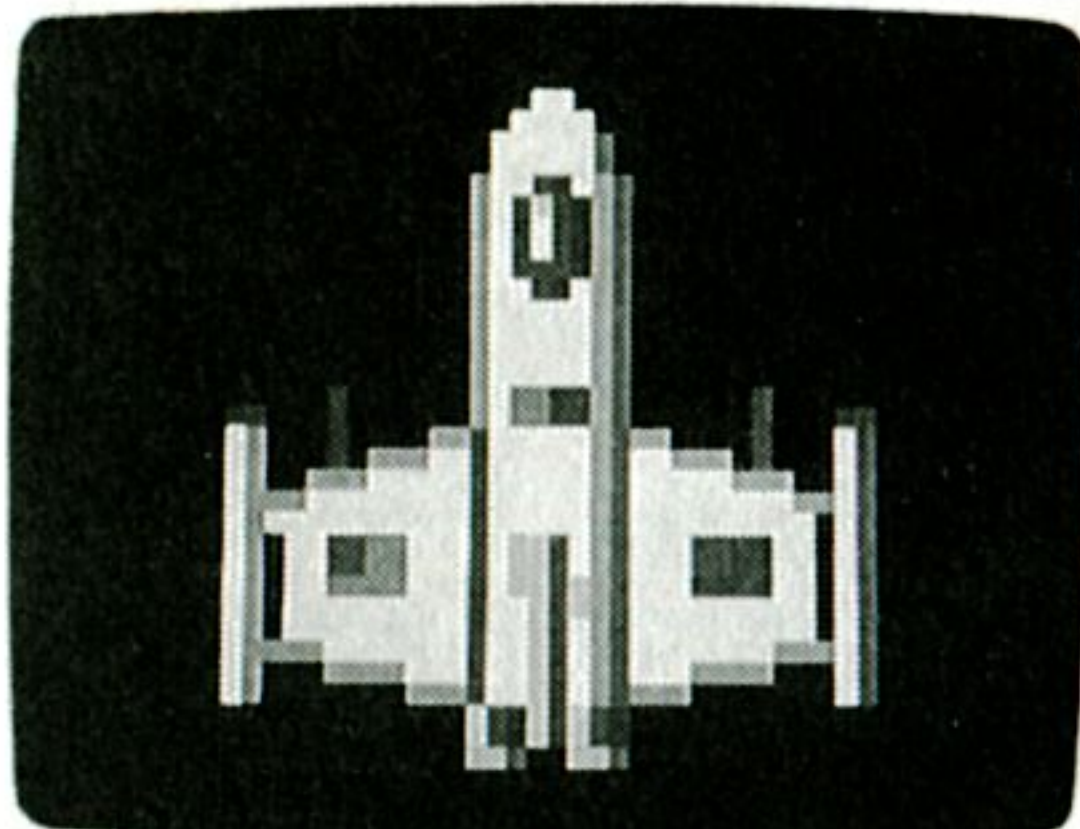
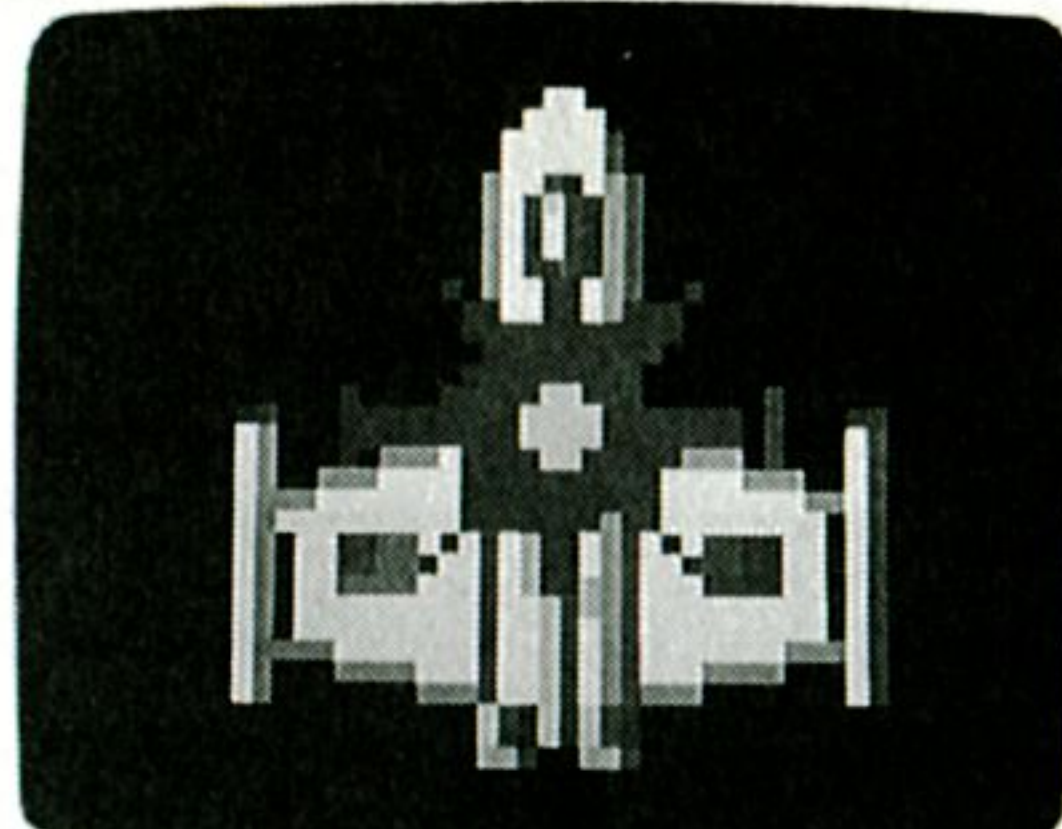
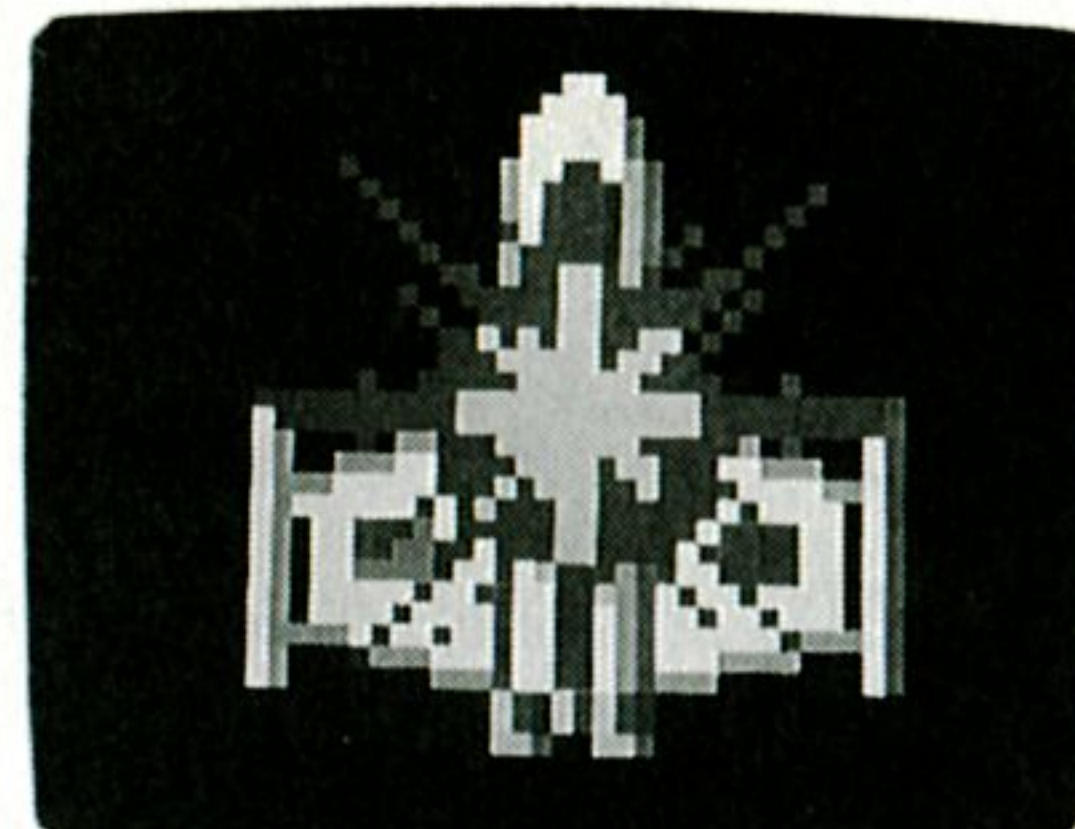
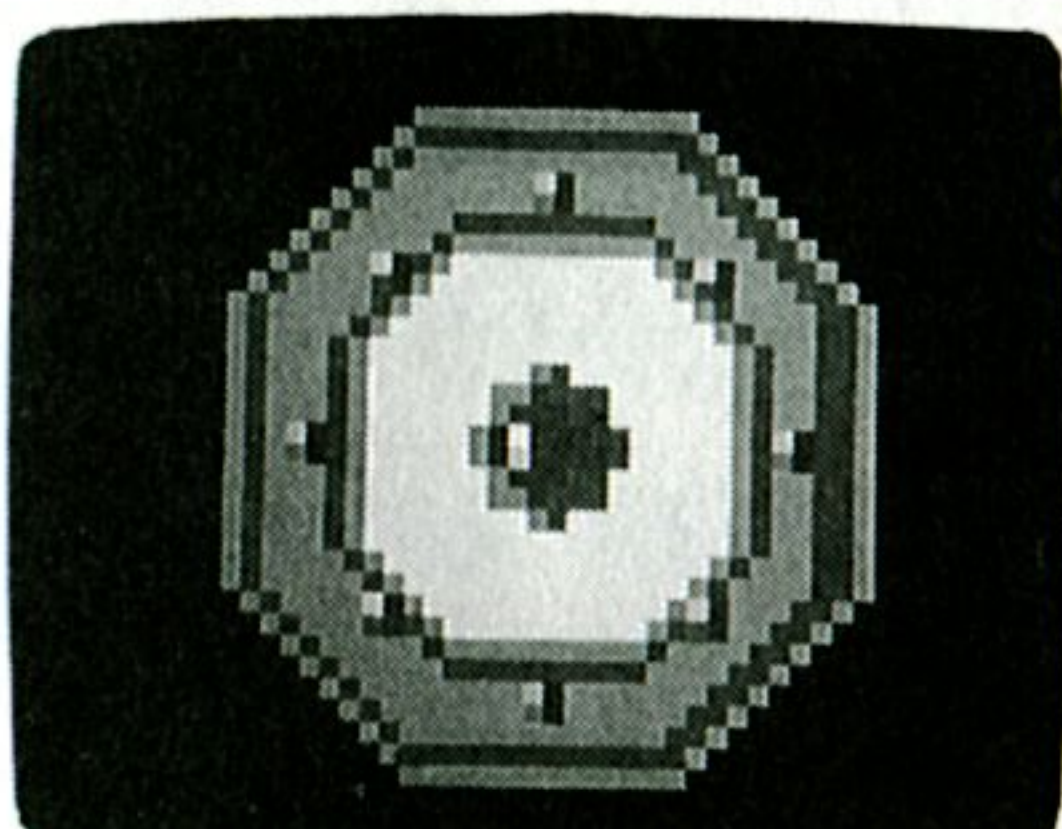
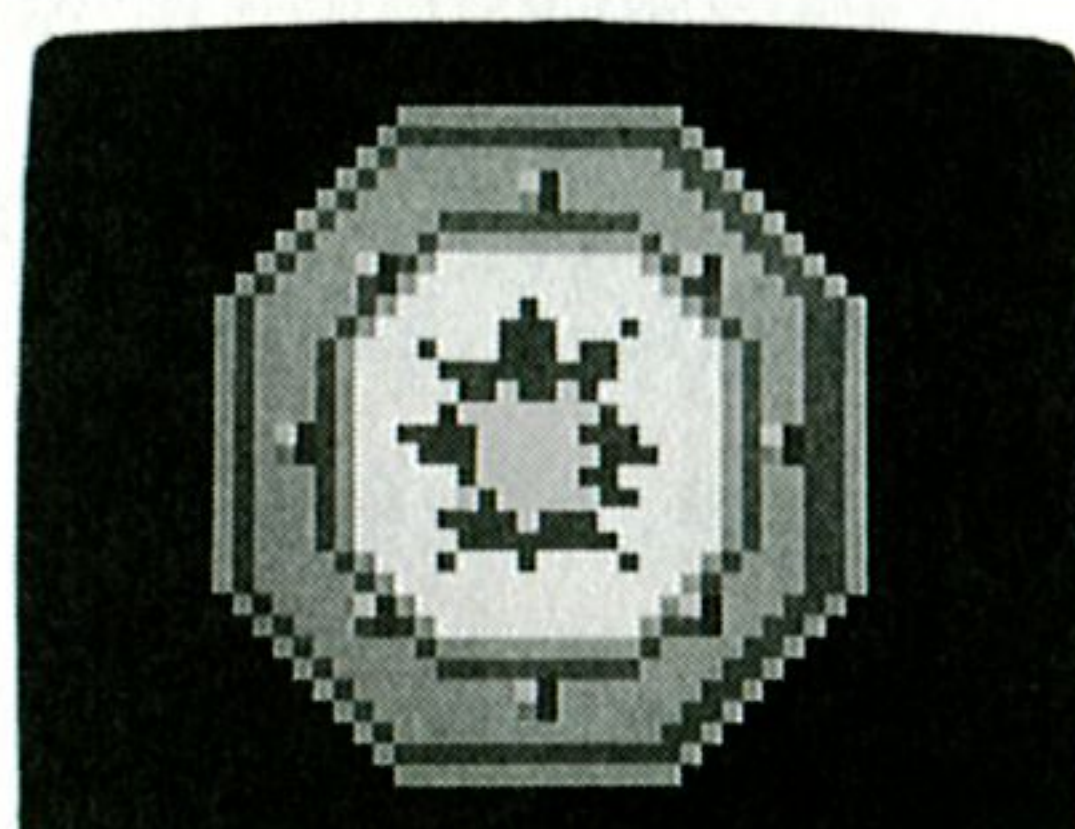
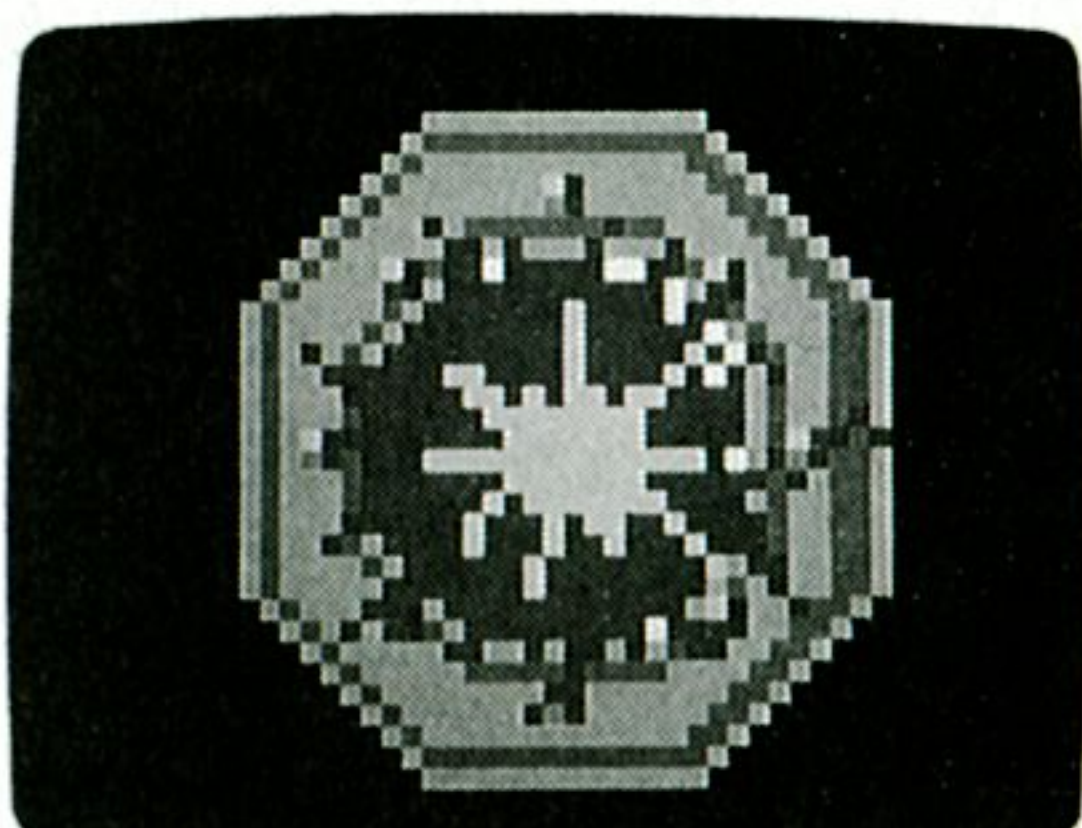
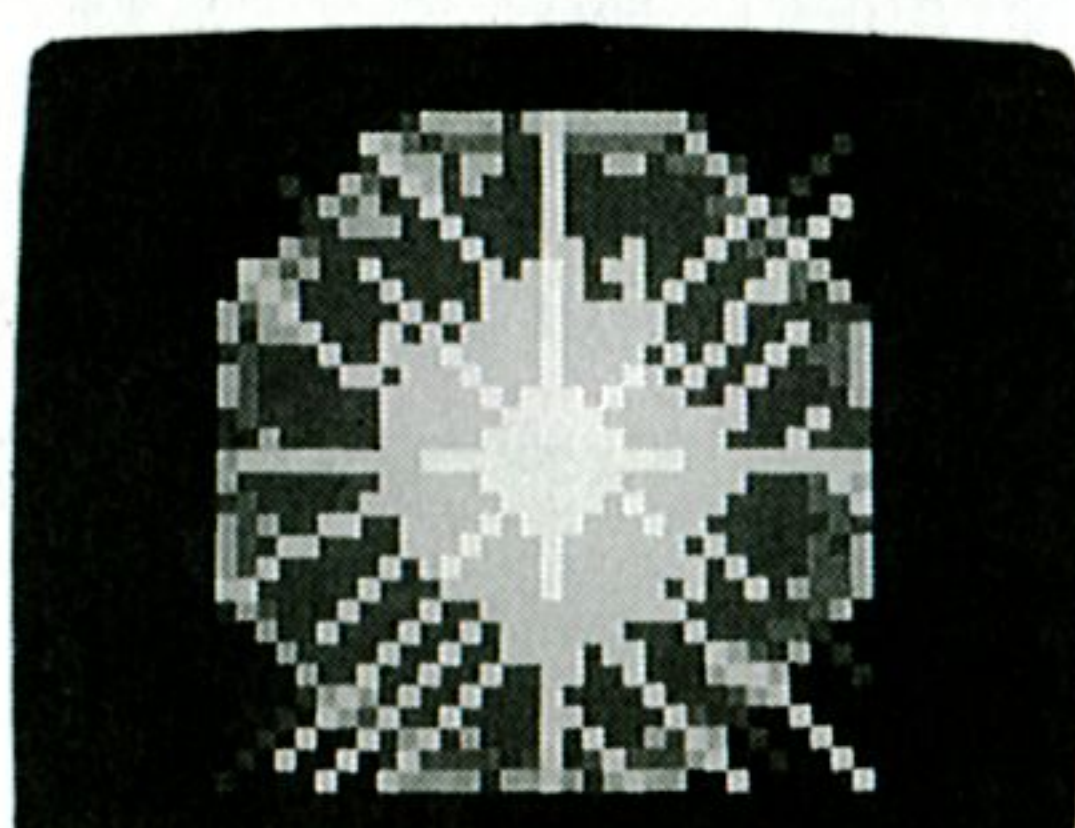
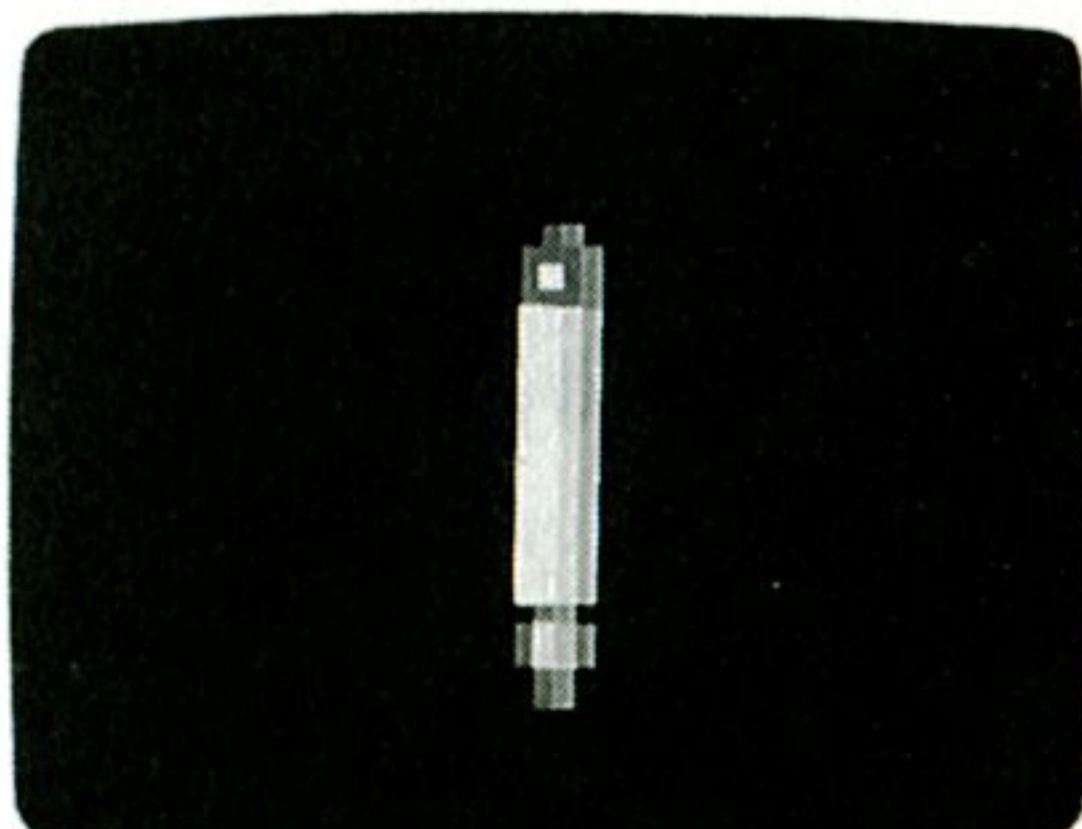
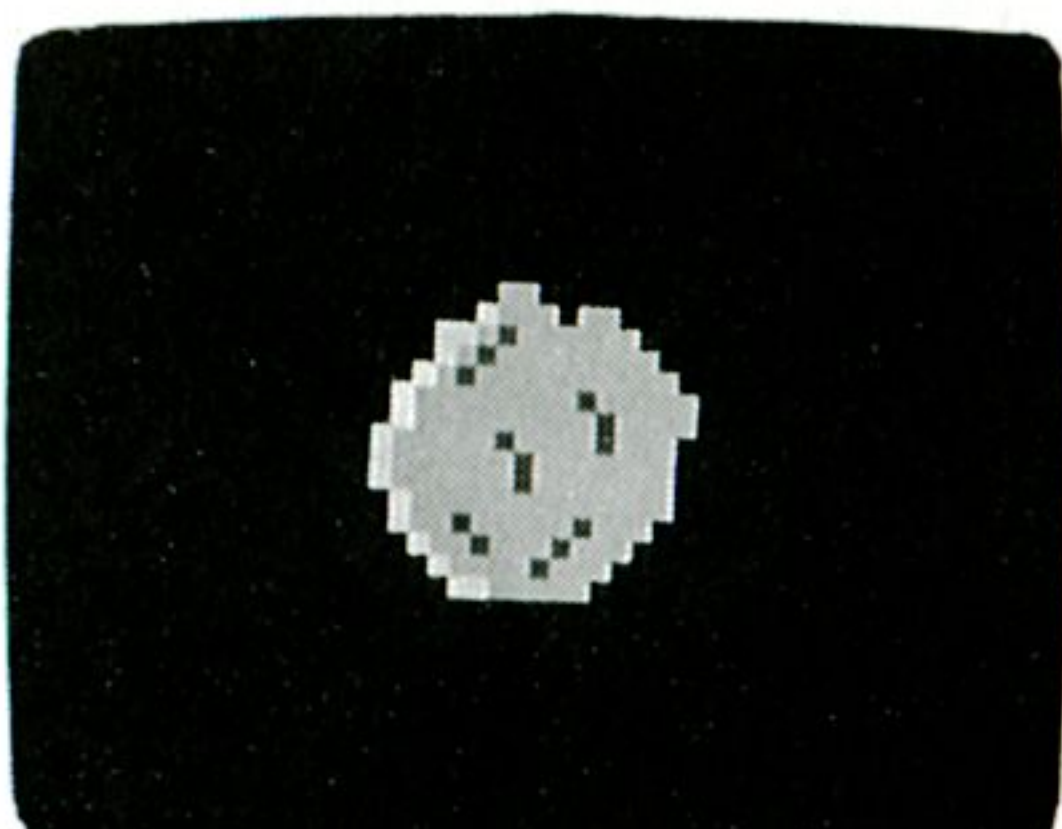
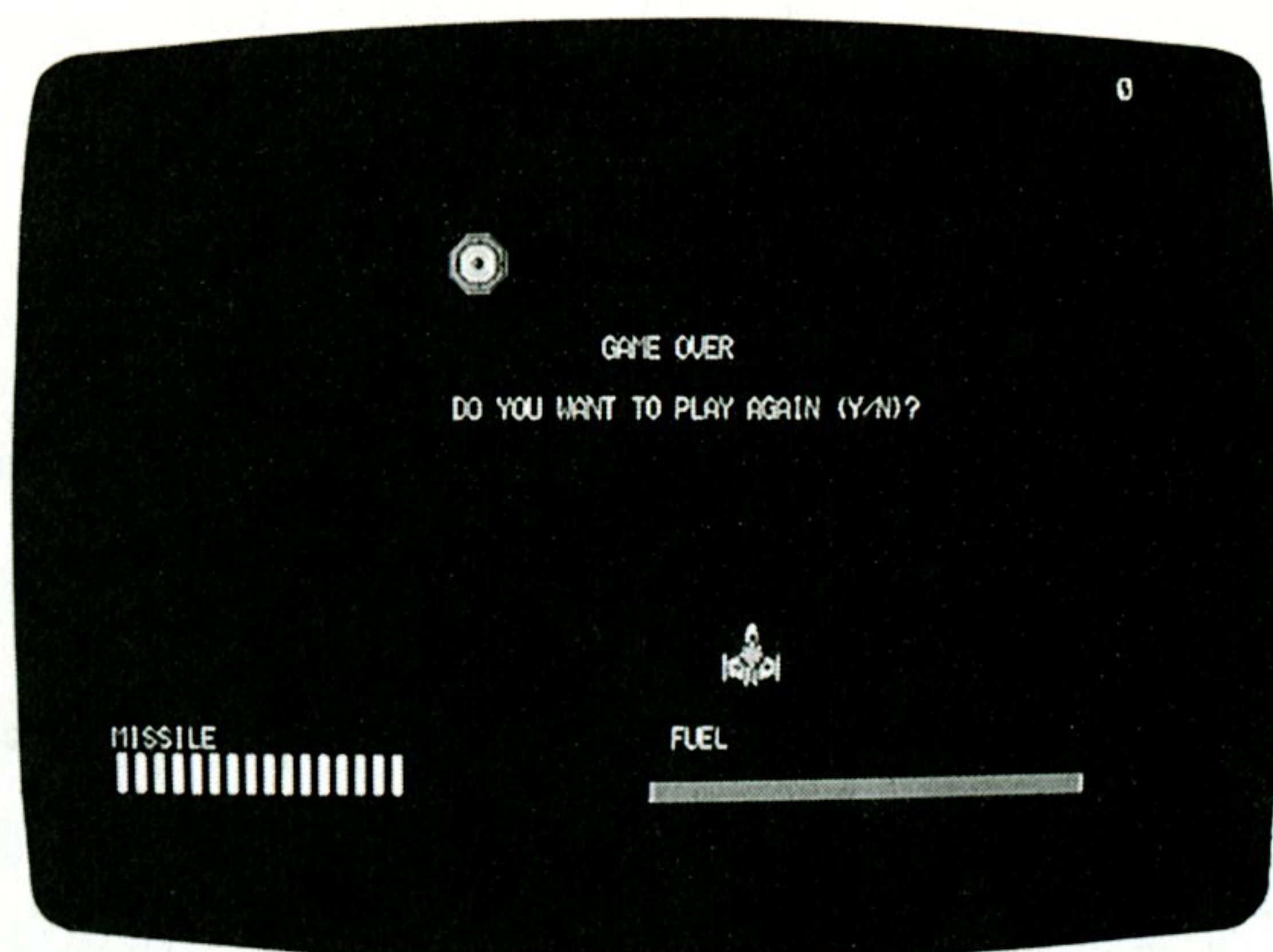
ではここで，本格的スクロールゲーム『SPACEWAR』に挑戦してみましょう．

時は2030年，宇宙人がUFOに乗り地球攻撃にと向ってきます．月の基地から発進した戦闘機に乗ったあなたは，UFOが地球攻撃ミサイルを発射するまでにやっつけなければいけません．ミサイルは16発，燃料にも限りがあります．隕石(いんせき)とUFOからの攻撃をかわしながら，あなたは任務を遂行できるか？

このゲームには，ゲーム作りのさまざまな秘密がいっぱい詰まっています．スルメを噛(か)むように良く味わってください．

ゲームが複雑になるにつれ，グラフィックがたくさんいるようになります．ゲームを作る前に，CGENE.BASを使って必要なグラフィックをかき，正しい名前でセーブしておいてください．まずUFOやFIGHTERの絵をかいてセーブしておき，次にその絵の上に火や爆破のようすを少しかき，それをBAKUHA1としてセーブし，さらに爆破を大きくしてそれをBAKUHA2としてセーブするようにすると比較的簡単にできます．

必要な絵は次のとおりです．

FIGHTER	UFO	MISSILE
MBAKUHA1	BAKUHA1	INSEKI
MBAKUHA2	BAKUHA2	TEKIFIRE
MBAKUHA3	BAKUHA3	
	BAKUHA4	

★カラー口絵
p. 3参照

『SPACEWAR』(シューティグゲーム)

★ゲームの遊び方

167ページのプログラム“SPACEWAR.BAS”をていねいに打ち込み，保存したあと，実行してください．

プログラム入力の前に，13ページのはじめの注意をよく読んでください．

数字の4で左，6で右に，8で上に，2で下に動きます．数字の0を押すと，味方のミサイルが発射されます．

隕石(いんせき)と敵のミサイルに当たると爆発してしまいます，くれぐれも当たらないように頑張ってください．1回の攻撃で5機以上のUFOをやっつけると，ミサイルと燃料が補給され，次の戦いが始まります．

1面では敵のUFOは攻撃してきませんが，2面からは敵も攻撃してきます．

★プログラムの解説

プログラムを上から順に見ていっても，全体像を理解するのは，なかなか簡単ではありません．そこでここでは，とくに大事な機能やテクニックを個別に見て行くようにしたいと思います．

●UFOの動き

敵のUFOは，右かと思えばまた左と，ヒラヒラと逃げまわります．時にはミサイルの当たる直前に身をかわしたりします．これは，次のようなやり方でプログラムされています．

毎回の位置をRNDで少しずつ変えようとすると，同じような位置に留まってしまってうまくいきません．ここでは，次のように行っています．

まず，SPACEWAR. BAS本体の169ページの8行目，次のように書かれている部分です．

```
SPACE1．BASプログラム
  UFOLIFE = 1
  UFOYOKO = 60 + INT(RND * 500): UFOTATE = 32 + INT(RND * 50)
  UFOMOKUTEKIY = 60 + INT(RND * 500): UFOMOKUTEKIX = 32 + INT(RND * 100)
```

ここで，最初に出現する位置をUFOYOKOおよびUFOTATEとしてRND関数を使って決めています．UFOYOKOの場合，PUTする位置が60から最大560の間になるように定めています．そして，出現する位置のほかに，目的地を同じようにして横方向UFOMOKUTEKIY，縦方向UFOMOKUTEKIXとして決めています．

次に，プログラムの下記の部分で，もし目的地の値の方が現在の位置の値より大きい場合は，現在位置の値を増やすように，少ない場合には現在位置の値を減らすように動くようにしています．

```
SPACE2．BASプログラム
  IF UFOYOKO < UFOMOKUTEKIY THEN UFOYOKO = UFOYOKO + 10 + UFOSOKUDO + UFOSOKUDO2
  IF UFOYOKO > UFOMOKUTEKIY THEN UFOYOKO = UFOYOKO - 10 - UFOSOKUDO - UFOSOKUDO2
  IF UFOTATE < UFOMOKUTEKIX THEN UFOTATE = UFOTATE + 4
  IF UFOTATE > UFOMOKUTEKIX THEN UFOTATE = UFOTATE - 4
```

(SPACEWAR.BASでは169ページの下から14行目になります)

また，ここでUFOSOKUDOという値を入れ，この速度自体をRND関数を使って変えることにより移動する速度が同じになって，動きが読めてしまうのを防いでいます．このようにして，現在位置を目的位置に近づけて行くのですが，このままでは現在位置と目的位置がほぼ一致したところでUFOが止まってしまいます．そこで，次のようなしかけを入れます．

```
SPACE3．BASプログラム
  IF ABS(UFOMOKUTEKIX - UFOTATE) < 4 THEN
  UFOMOKUTEKIX = 32 + INT(RND * 100)
  UFOSOKUDO = INT(RND * 8)
  END IF
  IF ABS(UFOMOKUTEKIY - UFOYOKO) < 21 THEN
   UFOMOKUTEKIY = 10 + INT(RND * 600)
  UFOSOKUDO = INT(RND * 8)
  END IF
```

(SPACEWAR.BAS では169ページ31行目になります)

これにより，UFO 現在位置が目的位置に近づいて来て，一定の範囲内に来ると目的位置そのものを別の値に変えています．またこのとき，一緒に UFOSOKUDO も 0 ～ 8 までの間で変化させています．

●ミサイル発射

ミサイルの発射をコントロールしているのは，プログラム SPACEWAR.BAS 本体の170ページ 5 行目以下の部分です．

```
SPACE4．BASプログラム
  IF FIRE = 0 THEN
     IF A$ = "0" THEN
     FIRE = 1
     MYOKOICHI = YOKO: MTATEICHI = TATE - 32
     LINE (MISSILEKAZU * 10, 370)-(MISSILEKAZU * 10 + 4, 390), 0, BF
     MISSILEKAZU = MISSILEKAZU - 1
     END IF
  END IF
```

この部分で重要なのは，最初の IF_FIRE_=_0_THEN の部分です．このプログラムでは，ミサイルが発射されているかどうかはFIREという変数をフラッグに使って，発射されていれば FIRE_=_1 とし，発射されていなければ FIRE_=_0 としています．ここで，最初のIF文で，いま現在 FIRE_=_0 つまり発射されたミサイルが飛んでいないときのみ，この部分のプログラムを実行するようにしています．

次に，IF_A$_=_"0" で， A$_=_INKEY$ としてキーボードから入力された文字が 0 のときにミサイル発射となるようにしています．次の行の FIRE_=_1 で，現在ミサイルが発射されていることをフラッグを立てて示し

ています．そして，MYOKOICHI_=_YOKO　としてミサイルの横位置を現在の戦闘機の横位置に一致させています．以後，このままの値を使うので，戦闘機の位置をどのように変えてもミサイルの横方向の位置には影響しません．また，　MTATEICHI_=_TATE_−_32　でミサイルを戦闘機の直前から発射させるようにしています．

そして，ミサイル数をそれまでの値から１つ減じています．

このように，この部分はミサイルの発射のみを扱い，実際のミサイルの動きはFIREが１かどうかを調べて別の部分で行っています．

また，一度ミサイルが発射されると，ミサイルが飛んでいってしまうか，UFOに当たって爆発するかして　FIRE_=_0　とされるまでは，この部分は0を押しても実行されません．

実際のミサイルの動きと衝突判定は，次のように行っています．

ＳＰＡＣＥ５．ＢＡＳプログラム

```
IF FIRE = 1 THEN
MTATEICHI = MTATEICHI - 20
IF MTATEICHI < 1 THEN FIRE = 0
IF F < 0 THEN F = 0
IF MTATEICHI > 1 THEN PUT (MYOKOICHI, MTATEICHI), MISSILE%
E = MYOKOICHI: F = MTATEICHI
END IF
IF ABS(MYOKOICHI - UFOYOKO) < 24 AND ABS(MTATEICHI - UFOTATE) < 24 THEN
GOSUB TEKIBAKUHA
END IF
```

(SPACEWAR. BAS 本体170ページ30行目から)

まず，IF_FIRE_=_1_THEN　の文で，FIREが1，つまりミサイルが発射されている場合のみ，この部分を実行させるようにしています．そして次に，

MTATEICHI_=_MTATEICHI_−_20

でミサイルの縦の位置が，このルーチンを通るたびに20ドット上に移動するようにしています．また次の行

IF_MTATEICHI_＜_1_THEN_FIRE_=_0

の行で，ミサイルが画面の上に飛び去る位置に来たときに，そのミサイルはなくなったことにしています．

その後でミサイルがまだ飛んでいる位置にあれば，新しい位置にミサイルの絵をPUTしています．そして，

IF_ABS(MYOKOICHI_－_UFOYOKO)_<_24_AND_ABS_(MTATEICHI_－_UFOTATE)_<_24_THEN
GOSUB_TEKIBAKUHA
END_IF

の行で，UFOとミサイルの衝突判定を行っています．つまり，UFOとミサイルの横位置が±24ドット以内に来て，かつそれと同時に縦位置が±24ドット以内に来たときにTEKIBAKUHAのサブルーチンに飛び，UFOを爆発させるようにしています．この数字を変えることにより，衝突の判定を甘くもからくもできます．

いまの位置は少し甘いのですが，近接信管(近づくと爆発する)が付いているんだと思ってください(本当は自分が有利なように作っているのです？)．

●敵の爆発

画面上で，敵がハデに爆発すると「ヤッター」という気になる人は多いと思います．そこで，爆発シーンをどう作るのかを説明しましょう．

爆発シーンの原理は簡単です．正常な絵から少しずつ爆発がひどくなって行く絵を順番に表示し，１つ１つの絵の間に若干の時間の遅れを入れることによって，爆発がひどくなって行くようすを示すことができます．ではプログラムを見て見ましょう．

ＳＰＡＣＥ６．ＢＡＳプログラム

```
TEKIBAKUHA:
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA1%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA2%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
```

```
PUT (YOKO, TATE), FIGHTER%, PSET
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA3%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA4%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
FIRE = 0
UFOLIFE = 0
MYOKOIHI = YOKO: MTATEICHI = TATE
SCORE = SCORE + 10000
TIMECOUNT = 0
RETURN
```

まず，GOSUB_GAMEN で画面の切り替えルーチンに飛んでいます(SPACEWAR 本体では170ページ下から５行目になります)．これで，１面をかいているときに０面を表示，０面をかいているときに１面を表示することができ，絵がきれいに出せます．

次に，３行目で UFO のいた位置に前もってかいてあった BAKUHA1%の絵を PUT しています．これにより，ほんの少し爆発しかけた絵を表示することができます．次の２行，

```
FOR_I_=_1_TO_BAKUHADELAY
NEXT
```

で，時間かせぎをしています．このような場合，直接数字を１つ１つ変えるよりも，あらかじめ BAKUHADELAY_=_500 のように１ヵ所で指定しておくとあとで実験しながら数値を変えるとき楽です．また，単なる時間かせぎではなく，この部分で PLAY 命令，SOUND 命令を使って爆発の音を出せば，より迫力のあるゲームになります．ぜひ挑戦してみてください．

次の行の GOSUB_HOSHI と，その次の行の GOSUB_INSEKI ですが，これは爆発のシーンの間，周りの星や隕石(いんせき)の動きが止まってはおかしいので入

れてあります．このように，各機能をサブルーチンで分けておくと，いろいろなところから呼び出せるので，プログラムが楽になります．あとは絵を少しずつハデな爆発になるように変えながら同じことを繰り返します．

これで，爆発のプログラム部分は終わりですが，このあとでいろいろ残った処理をしています．

ミサイルもUFOもなくなったので，FIRE_=_0,_UFOLIFE_=_0として，次のUFOの出現とミサイルの発射にそなえるようフラッグをリセットしておきます．

また，UFOが飛んでもいないミサイルと衝突判定されてしまうのを防ぐため，ミサイルの横位置と縦位置であるMYOKOICHIとMTATEICHIをいまの戦闘機の位置に合わせています．このあと，点数に10000点をプラスしています．

最後のTIMECOUNT_=_0は，UFOがあまりにすぐにまた出現しないようにするためのカウンタTIMECOUNTの値をリセットしています．

(プログラムを完全に入力したにもかかわらず動作しない場合は，11ページの囲み，および90ページのコラムを参考にしてみてください)

SPACEWAR．BASプログラム

```
SCREEN 88, 3, 1, 1
WINDOW SCREEN (0, 0)-(639, 399)
CLS
COL = 15
WIDTH 80, 25
RANDOMIZE TIMER
 SCORE = 0
 OLDSCORE = 0
 FUEL = 300: FUEL2 = FUEL
 MISSILEKAZU = 16
 ROUND = 0
 YOKO = 340: TATE = 300
 BAKUHADELAY = 500
 IDOURYOU = 20
 HOSHIIDOU = 6
 HA = 300
 HB = 50
 HC = 180
 HD = 251
 HE = 65
 HF = 115
 HG = 320
 HH = 70
 HI = 260
 HJ = 90
```

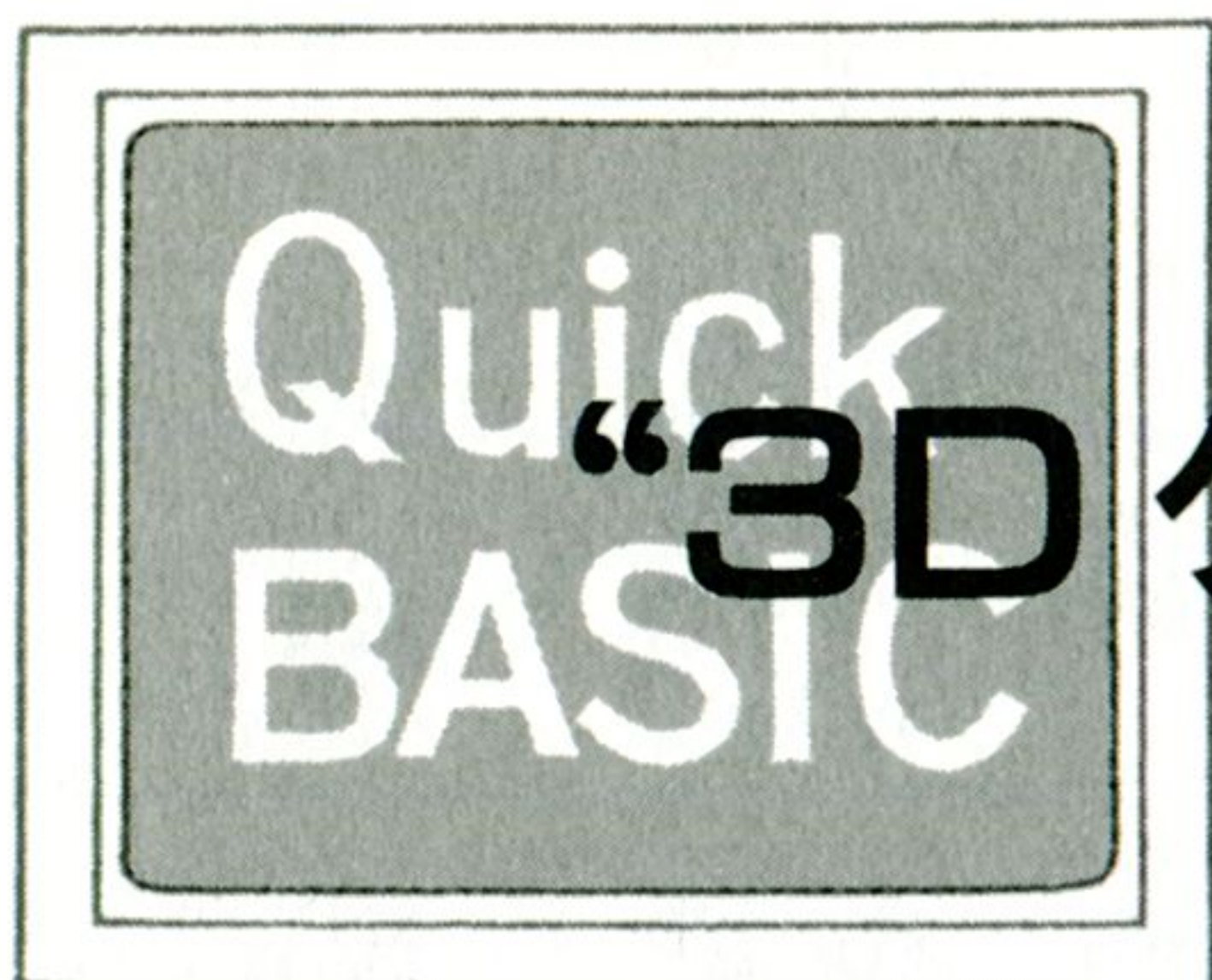
```
 INSEKITATEB = 200
 DIM SHOUKYO%(800)
 GET (1, 1)-(32, 32), SHOUKYO%
 DIM FIGHTER%(800)
 DIM UFO%(800)
 DIM MISSILE%(800)
 DIM BAKUHA1%(800)
 DIM BAKUHA2%(800)
 DIM BAKUHA3%(800)
 DIM BAKUHA4%(800)
 DIM INSEKI%(800)
 DIM MBAKUHA1%(800)
 DIM MBAKUHA2%(800)
 DIM MBAKUHA3%(800)
 DIM TEKIMISSILE%(800)
DEF SEG = VARSEG(FIGHTER%(0))
OFFSET% = VARPTR(FIGHTER%(0))
BLOAD "FIGHTER.GRA", OFFSET%
PUT (340, 300), FIGHTER%
DEF SEG = VARSEG(UFO%(0))
OFFSET% = VARPTR(UFO%(0))
BLOAD "UFO.GRA", OFFSET%
DEF SEG = VARSEG(MISSILE%(0))
OFFSET% = VARPTR(MISSILE%(0))
BLOAD "MISSILE.GRA", OFFSET%
DEF SEG = VARSEG(BAKUHA1%(0))
OFFSET% = VARPTR(BAKUHA1%(0))
BLOAD "BAKUHA1.GRA", OFFSET%
DEF SEG = VARSEG(BAKUHA2%(0))
OFFSET% = VARPTR(BAKUHA2%(0))
BLOAD "BAKUHA2.GRA", OFFSET%
DEF SEG = VARSEG(BAKUHA3%(0))
OFFSET% = VARPTR(BAKUHA3%(0))
BLOAD "BAKUHA3.GRA", OFFSET%
DEF SEG = VARSEG(BAKUHA4%(0))
OFFSET% = VARPTR(BAKUHA4%(0))
BLOAD "BAKUHA4.GRA", OFFSET%
DEF SEG = VARSEG(INSEKI%(0))
OFFSET% = VARPTR(INSEKI%(0))
BLOAD "INSEKI.GRA", OFFSET%
DEF SEG = VARSEG(MBAKUHA1%(0))
OFFSET% = VARPTR(MBAKUHA1%(0))
BLOAD "MBAKUHA1.GRA", OFFSET%
DEF SEG = VARSEG(MBAKUHA2%(0))
OFFSET% = VARPTR(MBAKUHA2%(0))
BLOAD "MBAKUHA2.GRA", OFFSET%
DEF SEG = VARSEG(MBAKUHA3%(0))
OFFSET% = VARPTR(MBAKUHA3%(0))
BLOAD "MBAKUHA3.GRA", OFFSET%
DEF SEG = VARSEG(TEKIMISSILE%(0))
OFFSET% = VARPTR(TEKIMISSILE%(0))
BLOAD "TEKIFIRE.GRA", OFFSET%
DEF SEG
```

```
 LOCATE 23, 2: PRINT "MISSILE"
 FOR I = 1 TO MISSILEKAZU
 LINE (I * 10, 370)-(I * 10 + 4, 390), 7, BF
 NEXT I
 LOCATE 23, 40: PRINT "FUEL"
 LINE (300, 385)-(300 + FUEL, 395), 2, BF
 UFOSOKUDO = ABS(INT(RND * 8))
  UFOLIFE = 1
  UFOYOKO = 60 + INT(RND * 500): UFOTATE = 32 + INT(RND * 50)
  UFOMOKUTEKIY = 60 + INT(RND * 500): UFOMOKUTEKIX = 32 + INT(RND * 100)
 START:
 GOSUB GAMEN
 UFOSOKUDO2 = INT(RND * 2)
 LOCATE 23, 2: PRINT "MISSILE"
 LOCATE 23, 40: PRINT "FUEL"
 GOSUB HOSHI
 GOSUB INSEKI
 IF ABS(INSEKIYOKOA - YOKO) < 16 AND ABS(INSEKITATEA - TATE) < 16 THEN
 GOSUB MIKATABAKUHATSU
 END IF
 IF ABS(INSEKIYOKOB - YOKO) < 16 AND ABS(INSEKITATEB - TATE) < 16 THEN
 GOSUB MIKATABAKUHATSU
 END IF
 TIMECOUNT = TIMECOUNT + 1
 IF TIMECOUNT > 30 THEN TIMECOUNT = 0
 IF UFOLIFE = 0 AND TIMECOUNT > 20 THEN
 UFOYOKO = 60 + INT(RND * 500): UFOTATE = 32 + INT(RND50)
  C = UFOYOKO: D = UFOTATE
  UFOLIFE = 1
  END IF
 IF ABS(UFOMOKUTEKIX - UFOTATE) < 4 THEN
 UFOMOKUTEKIX = 32 + INT(RND * 100)
 UFOSOKUDO = INT(RND * 8)
 END IF
 IF ABS(UFOMOKUTEKIY - UFOYOKO) < 21 THEN
  UFOMOKUTEKIY = 10 + INT(RND * 600)
 UFOSOKUDO = INT(RND * 8)
 END IF
 IF UFOYOKO < UFOMOKUTEKIY THEN UFOYOKO = UFOYOKO + 10 + UFOSOKUDO + UFOSOKUDO2
 IF UFOYOKO > UFOMOKUTEKIY THEN UFOYOKO = UFOYOKO - 10 - UFOSOKUDO - UFOSOKUDO2
 IF UFOTATE < UFOMOKUTEKIX THEN UFOTATE = UFOTATE + 4
 IF UFOTATE > UFOMOKUTEKIX THEN UFOTATE = UFOTATE - 4
 IF UFOYOKO < 1 THEN UFOYOKO = 607
 IF UFOYOKO > 607 THEN UFOYOKO = 1
 IF UFOLIFE = 1 THEN PUT (UFOYOKO, UFOTATE), UFO%, PSET
 C = UFOYOKO: D = UFOTATE
 A$ = INKEY$
    IF A$ = "1" THEN YOKO = YOKO - IDOURYOU: TATE = TATE + IDOURYOU
    IF A$ = "2" THEN TATE = TATE + IDOURYOU
    IF A$ = "3" THEN YOKO = YOKO + IDOURYOU: TATE = TATE + IDOURYOU
    IF A$ = "4" THEN YOKO = YOKO - IDOURYOU
    IF A$ = "6" THEN YOKO = YOKO + IDOURYOU
```

```
    IF A$ = "7" THEN YOKO = YOKO - IDOURYOU: TATE = TATE - IDOURYOU
    IF A$ = "8" THEN TATE = TATE - IDOURYOU
    IF A$ = "9" THEN YOKO = YOKO + IDOURYOU: TATE = TATE - IDOURYOU
 IF MISSILEKAZU <= 0 AND FIRE = 0 THEN GOTO CHECKOWARI
 IF FIRE = 0 THEN
    IF A$ = "0" THEN
    FIRE = 1
    MYOKOICHI = YOKO: MTATEICHI = TATE - 32
    LINE (MISSILEKAZU * 10, 370)-(MISSILEKAZU * 10 + 4, 390), 0, BF
    MISSILEKAZU = MISSILEKAZU - 1
    END IF
 END IF
 IF A$ = "Q" THEN GOTO OWARI
 IF A$ = "q" THEN GOTO OWARI
 IF YOKO > 607 THEN YOKO = 1
 IF YOKO < 1 THEN YOKO = 607
 IF TATE > 310 THEN TATE = 310
 IF TATE < 270 THEN TATE = 270
 PUT (YOKO, TATE), FIGHTER%, PSET
     A = YOKO: B = TATE
 IF ABS(INSEKIYOKOA - YOKO) < 16 AND ABS(INSEKITATEA - TATE) < 16 THEN
 GOSUB MIKATABAKUHATSU
 END IF
 IF ABS(INSEKIYOKOB - YOKO) < 16 AND ABS(INSEKITATEB - TATE) < 16 THEN
 GOSUB MIKATABAKUHATSU
 END IF
 IF ABS(TEKIMYOKOICHI - YOKO) < 16 AND ABS(TEKIMTATEICHI - TATE) < 16 THEN
 GOSUB MIKATABAKUHATSU
 END IF
 IF FIRE = 1 THEN
 MTATEICHI = MTATEICHI - 20
 IF MTATEICHI < 1 THEN FIRE = 0
 IF F < 0 THEN F = 0
 IF F < 360 THEN PUT (E, F), SHOUKYO%, PSET
 IF MTATEICHI > 1 THEN PUT (MYOKOICHI, MTATEICHI), MISSILE%, OR
 E = MYOKOICHI: F = MTATEICHI
 END IF
 IF ABS(MYOKOICHI - UFOYOKO) < 24 AND ABS(MTATEICHI - UFOTATE) < 24 THEN
 GOSUB TEKIBAKUHA
 END IF
 LOCATE 1, 70: PRINT SCORE
 FUEL2 = FUEL2 - .2
 FUEL = INT(FUEL2)
 LINE (600, 385)-(300 + FUEL, 395), 0, BF
 IF FUEL <= 0 THEN GOTO CHECKOWARI
 IF ROUND > 0 THEN GOSUB TEKIKOUGEKIA
 GOSUB TEKIKOUGEKIB
 GOTO START:
 TEKIBAKUHA:
 GOSUB GAMEN
 PUT (UFOYOKO, UFOTATE), BAKUHA1%, PSET
 FOR I = 1 TO BAKUHADELAY
 NEXT
```

```
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA2%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA3%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
GOSUB GAMEN
PUT (UFOYOKO, UFOTATE), BAKUHA4%, PSET
FOR I = 1 TO BAKUHADELAY
NEXT
GOSUB HOSHI
GOSUB INSEKI
PUT (YOKO, TATE), FIGHTER%, PSET
FIRE = 0
UFOLIFE = 0
MYOKOIHI = YOKO: MTATEICHI = TATE
SCORE = SCORE + 10000
TIMECOUNT = 0
RETURN
OWARI:
LOCATE 10, 35: PRINT "GAME OVER"
LOCATE 12, 25: PRINT "DO YOU WANT TO PLAY AGAIN (Y/N)?"
300
A$ = INKEY$
IF A$ = "" THEN GOTO 300
IF A$ = "y" THEN RUN
IF A$ = "Y" THEN RUN
IF A$ = "ﾝ" THEN RUN
END
HOSHI:
HA = HA + HOSHIIDOU
IF HA > 330 THEN HA = 0
 PSET (0, HA), 7
HB = HB + HOSHIIDOU
IF HB > 330 THEN HB = 0
 PSET (100, HB), 7
HC = HC + HOSHIIDOU
IF HC > 330 THEN HC = 0
 PSET (150, HC), 7
HD = HD + HOSHIIDOU
IF HD > 330 THEN HD = 0
 PSET (200, HD), 7
```

```
HE = HE + HOSHIIDOU
IF HE > 330 THEN HE = 0
 PSET (250, HE), 7
HF = HF + HOSHIIDOU
IF HF > 330 THEN HF = 0
 PSET (300, HF), 7
HG = HG + HOSHIIDOU
IF HG > 330 THEN HG = 0
 PSET (350, HG), 7
HH = HH + HOSHIIDOU
IF HH > 330 THEN HH = 0
 PSET (400, HH), 7
HI = HI + HOSHIIDOU
IF HI > 330 THEN HI = 0
 PSET (520, HI), 7
HJ = HJ + HOSHIIDOU
IF HJ > 330 THEN HJ = 0
 PSET (600, HJ), 7
HK = HK + HOSHIIDOU
IF HK > 330 THEN HK = 0
 PSET (614, HK), 7
HL = HL + HOSHIIDOU
IF HL > 330 THEN HL = 0
 PSET (430, HL), 7
RETURN
INSEKI:
IF INSEKILIFEA = 0 THEN
INSEKIYOKOA = INT(10 + RND * 580): INSEKILIFEA = 1
END IF
INSEKITATEA = INSEKITATEA + HOSHIIDOU
IF INSEKITATEA > 300 THEN
INSEKITATEA = 0
INSEKILIFEA = 0
END IF
IF INSEKILIFEA = 1 THEN PUT (INSEKIYOKOA, INSEKITATEA), INSEKI%, PSET
KINSEKIYA = INSEKIYOKOA: KINSEKITA = INSEKITATEA
IF INSEKILIFEB = 0 THEN
INSEKIYOKOB = INT(10 + RND * 580): INSEKILIFEB = 1
END IF
INSEKITATEB = INSEKITATEB + HOSHIIDOU
IF INSEKITATEB > 300 THEN
INSEKITATEB = 0
INSEKILIFEB = 0
END IF
IF INSEKILIFEB = 1 THEN PUT (INSEKIYOKOB, INSEKITATEB), INSEKI%, PSET
KINSEKIYB = INSEKIYOKOB: KINSEKITB = INSEKITATEB
RETURN
TEKIKOUGEKIA:
IF UFOLIFE = 0 THEN GOTO 30000
IF ABS(UFOYOKO - YOKO) < (20 + INT(RND * 10)) AND TEKIFIRE = 0 THEN
TEKIMYOKOICHI = UFOYOKO: TEKIMTATEICHI = UFOTATE + 32
IF INT(RND * 10) > 7 THEN TEKIFIRE = 1
END IF
```

```
30000
RETURN
TEKIKOUGEKIB:
IF TEKIFIRE = 1 THEN
TEKIMTATEICHI = TEKIMTATEICHI + 20
IF TEKIMTATEICHI > 330 THEN TEKIFIRE = 0
IF P > 1 AND Q < 360 THEN PUT (P, Q), SHOUKYO%, PSET
IF TEKIMTATEICHI < 330 THEN PUT (TEKIMYOKOICHI, TEKIMTATEICHI), TEKIMISSILE%
P = TEKIMYOKOICHI: Q = TEKIMTATEICHI
END IF
RETURN
MIKATABAKUHATSU:
PUT (YOKO, TATE), FIGHTER%, PSET
IF UFOLIFE = 1 THEN PUT (UFOYOKO, UFOTATE), UFO%, PSET
GOSUB GAMEN
PUT (YOKO, TATE), MBAKUHA1%, PSET
FOR I = 1 TO (BAKUHADELAY * 2 + 700)
NEXT
IF UFOLIFE = 1 THEN PUT (UFOYOKO, UFOTATE), UFO%, PSET
GOSUB GAMEN
PUT (YOKO, TATE), MBAKUHA2%, PSET
FOR I = 1 TO (BAKUHADELAY * 3 + 700)
NEXT
IF UFOLIFE = 1 THEN PUT (UFOYOKO, UFOTATE), UFO%, PSET
GOSUB GAMEN
PUT (YOKO, TATE), MBAKUHA3%, PSET
FOR I = 1 TO (BAKUHADELAY * 3 + 700)
NEXT
IF UFOLIFE = 1 THEN PUT (UFOYOKO, UFOTATE), UFO%, PSET
GOTO OWARI
RETURN
GAMEN:
IF GAMEN = 1 THEN SCREEN 88, 3, 0, 1
IF GAMEN = 0 THEN SCREEN 88, 3, 1, 0
CLS 1
GAMEN = GAMEN + 1
IF GAMEN > 1 THEN GAMEN = 0
FOR I = 1 TO MISSILEKAZU
LINE (I * 10, 370)-(I * 10 + 4, 390), 7, BF
NEXT I
LINE (300, 385)-(300 + FUEL, 395), 2, BF
RETURN
CHECKOWARI:
ROUND = ROUND + 1
IF (SCORE - OLDSCORE) > 40000 THEN
MISSILEKAZU = 16: FUEL = 300: FUEL2 = FUEL: OLDSCORE = SCORE
FOR I = 1 TO MISSILEKAZU
LINE (I * 10, 370)-(I * 10 + 4, 390), 7, BF
NEXT I
LINE (300, 385)-(300 + FUEL, 395), 2, BF
GOTO START:
END IF
GOTO OWARI
```

Quick BASIC "3D ゲームの大秘密"

セガは,『スペースハリアー』や『トップガン』というような敵が向うから向かって近づいて来る中を,自分が動き回る3次元的な立体ゲームを得意としています.

普通のゲームが上から見たところか,そうでなければ横方向から見た形のものが多いのに比べ,より自分がその中にいるという臨場感にあふれたゲームとなっています.

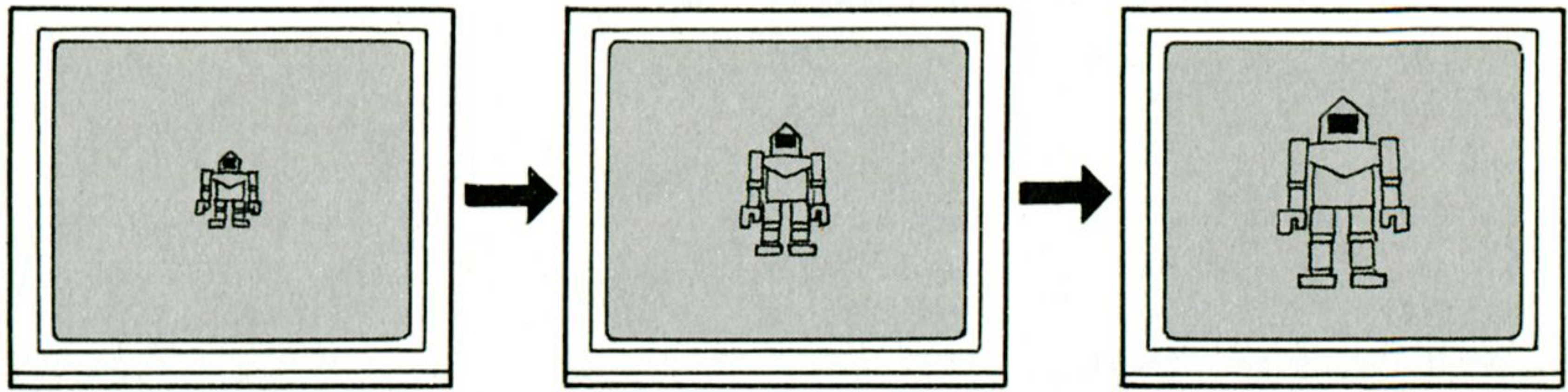
今度は,この3Dゲーム(3 dimension の略,三次元のゲーム)の秘密にせまって見ましょう(セガさん秘密をバラしてゴメンナサイ).

これらのゲームの絵は,2つのシステムから成り立っています.1つは,概略の景色などを計算により直線で分けて,その中をある色で平面として塗って行く方法です.このやり方により,概略の地形などを表示します.

次に,敵のキャラクタや障害物などが遠くからだんだんと迫ってくるように見える表示です.これは,実はキャラクタを画面にPUTしているのですが,同じキャラクタに対して小さなものから大きなものまで,複数の絵を用意しておきます.これらとの間の距離を計算して,遠くにあるときは非常に小さなキャラクタから始め,近づくにつれてどんどん表示する絵を大きなものに変えていっているのです.

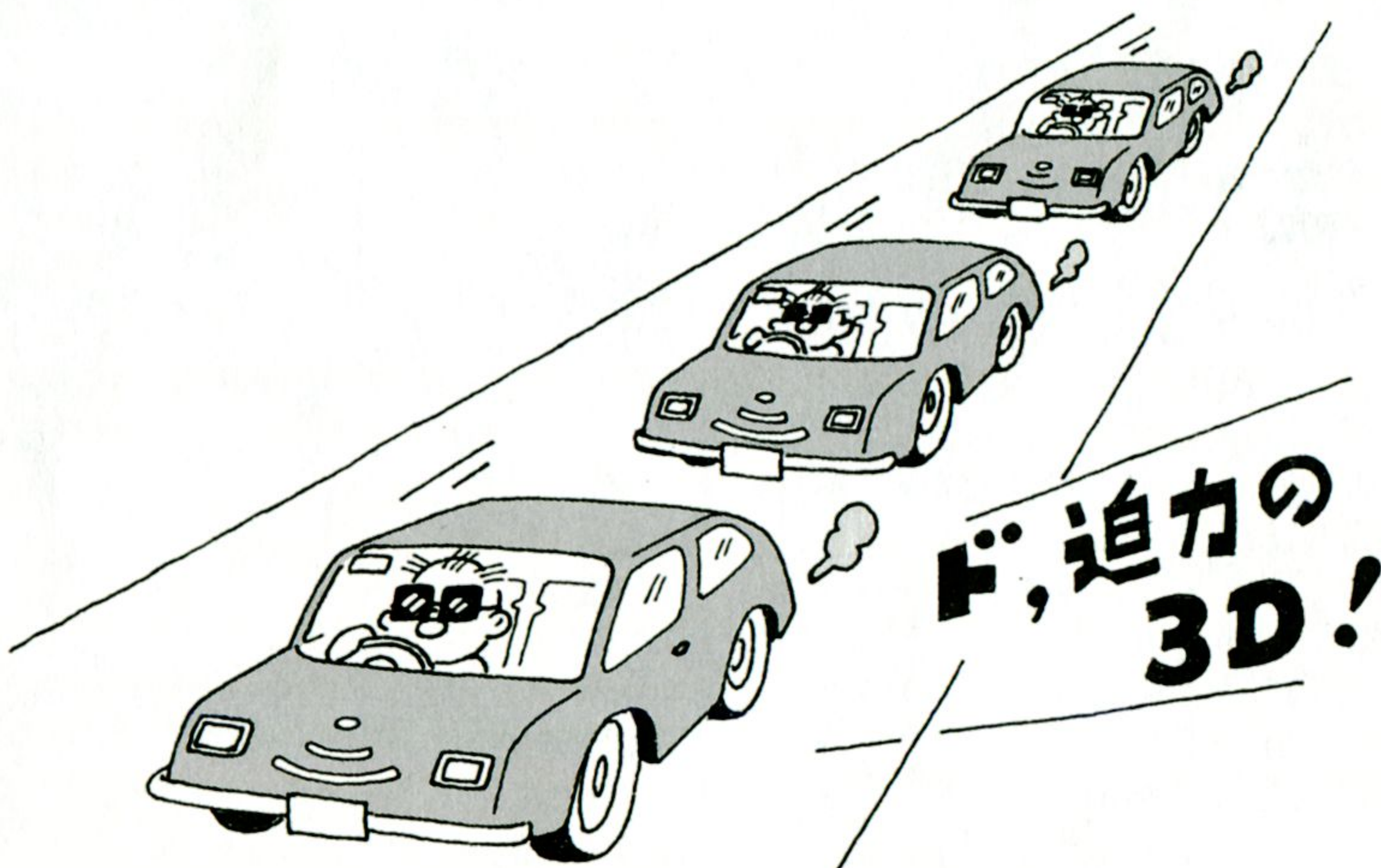
背景となる平面の変化の上に,だんだんと大きくなるキャラクタが動くことにより,立体的に迫って来る感じをうまく出しています.

絵を大きくするのに,1つずつ全部描いているのは大変です.そこで,絵を2倍,3倍,4倍の大きさにして,それぞれをセーブするプログラムEXPAND.BASを作って見ました.

まずキャラクタジェネレータ用プログラムCGENE.BASで敵のモビルスーツや怪獣の絵を描いてください.このとき32×32のと枠いっぱいに描かず上下2ドットずつ左右4ドットずつ程度あけておいてください.こうすることにより,消去のプロセスを減らすことができます.

オッ，迫力が出るなあ～
これが，3Dゲームの
ダイゴミだ！……

作ったキャラクタは，7文字以下の名前を付けてセーブしておいてください．CGENE.BASを抜けてプログラムEXPAND. BASを実行してください．ファイル名を聞いて来るので，先程セーブした名前を入れてください(拡張子.GRAはいりません)．

プログラムは，自動的に2倍，3倍，4倍の大きさの絵を作り，元の名前のうしろに2，3，4を付けた名前でセーブします．あとはこの大きくなったキャラクタを使い，最初は小さなキャラクタを表示し，だんだんと大きなキャラクタを表示するようにすれば，敵がせまってくるゲームを作ることができます．

ド，迫力の
3D！

EXPAND．BASプログラム

```
SCREEN 88, 3, 1, 1
CLS
WINDOW SCREEN (0, 0)-(639, 399)
DIM Z%(800)
START:
INPUT "FILENAME=", NAME$
IF NAME$ = "" THEN GOTO START:
IF NAME$ = "@" THEN END
DEF SEG = VARSEG(Z%(0))
OFFSET% = VARPTR(Z%(0))
BLOAD NAME$ + ".GRA", OFFSET%
PUT (500, 100), Z%
FOR I = 0 TO 31
FOR J = 0 TO 31
COLOR2 = POINT(500 + I, 100 + J)
 PSET (15 + I * 2, 15 + J * 2), COLOR2
 PSET (15 + I * 2 + 1, 15 + J * 2), COLOR2
 PSET (15 + I * 2, 15 + J * 2 + 1), COLOR2
 PSET (15 + I * 2 + 1, 15 + J * 2 + 1), COLOR2
NEXT J
NEXT I
LOCATE 13, 52: PRINT "      "
DIM Z2%(3200)
DEF SEG = VARSEG(Z2%(0))
GET (15, 15)-(15 + 32 * 2, 15 + 32 * 2), Z2%
OFFSET% = VARPTR(Z2%(0))
LOCATE 13, 52
BSAVE NAME$ + "2" + ".GRA", OFFSET%, 3200
LOCATE 13, 52: PRINT "      "
CLS
DEF SEG = VARSEG(Z%(0))
OFFSET% = VARPTR(Z%(0))
BLOAD NAME$ + ".GRA", OFFSET%
PUT (500, 100), Z%
FOR I = 0 TO 31
FOR J = 0 TO 31
COLOR2 = POINT(500 + I, 100 + J)
 PSET (15 + I * 3, 15 + J * 3), COLOR2
 PSET (15 + I * 3 + 1, 15 + J * 3), COLOR2
 PSET (15 + I * 3, 15 + J * 3 + 1), COLOR2
 PSET (15 + I * 3 + 1, 15 + J * 3 + 1), COLOR2
 PSET (15 + I * 3 + 2, 15 + J * 3), COLOR2
 PSET (15 + I * 3, 15 + J * 3 + 2), COLOR2
 PSET (15 + I * 3 + 2, 15 + J * 3 + 2), COLOR2
 PSET (15 + I * 3 + 2, 15 + J * 3 + 1), COLOR2
 PSET (15 + I * 3 + 1, 15 + J * 3 + 2), COLOR2
```

```
NEXT J
NEXT I
LOCATE 13, 52: PRINT "     "
LOCATE 13, 52: PRINT "     "
DIM Z3%(7200)
DEF SEG = VARSEG(Z3%(0))
GET (15, 15)-(15 + 32 * 3, 15 + 32 * 3), Z3%
OFFSET% = VARPTR(Z3%(0))
LOCATE 13, 52
BSAVE NAME$ + "3" + ".GRA", OFFSET%, 7200
CLS:
DEF SEG = VARSEG(Z%(0))
OFFSET% = VARPTR(Z%(0))
BLOAD NAME$ + ".GRA", OFFSET%
PUT (500, 100), Z%
FOR I = 0 TO 31
FOR J = 0 TO 31
COLOR2 = POINT(500 + I, 100 + J)
 PSET (15 + I * 4, 15 + J * 4), COLOR2
 PSET (15 + I * 4 + 1, 15 + J * 4), COLOR2
 PSET (15 + I * 4, 15 + J * 4 + 1), COLOR2
 PSET (15 + I * 4 + 1, 15 + J * 4 + 1), COLOR2
 PSET (15 + I * 4 + 2, 15 + J * 4), COLOR2
 PSET (15 + I * 4, 15 + J * 4 + 2), COLOR2
 PSET (15 + I * 4 + 2, 15 + J * 4 + 2), COLOR2
 PSET (15 + I * 4 + 2, 15 + J * 4 + 1), COLOR2
 PSET (15 + I * 4 + 1, 15 + J * 4 + 2), COLOR2
 PSET (15 + I * 4 + 3, 15 + J * 4), COLOR2
 PSET (15 + I * 4, 15 + J * 4 + 3), COLOR2
 PSET (15 + I * 4 + 3, 15 + J * 4 + 1), COLOR2
 PSET (15 + I * 4 + 3, 15 + J * 4 + 2), COLOR2
 PSET (15 + I * 4 + 3, 15 + J * 4 + 3), COLOR2
 PSET (15 + I * 4 + 1, 15 + J * 4 + 3), COLOR2
 PSET (15 + I * 4 + 2, 15 + J * 4 + 3), COLOR2
NEXT J
NEXT I
LOCATE 13, 52: PRINT "     "
DIM Z4%(12800)
DEF SEG = VARSEG(Z4%(0))
GET (15, 15)-(15 + 32 * 4, 15 + 32 * 4), Z4%
OFFSET% = VARPTR(Z4%(0))
BSAVE NAME$ + "4" + ".GRA", OFFSET%, 12800
LOCATE 13, 52: PRINT "     "
END
```

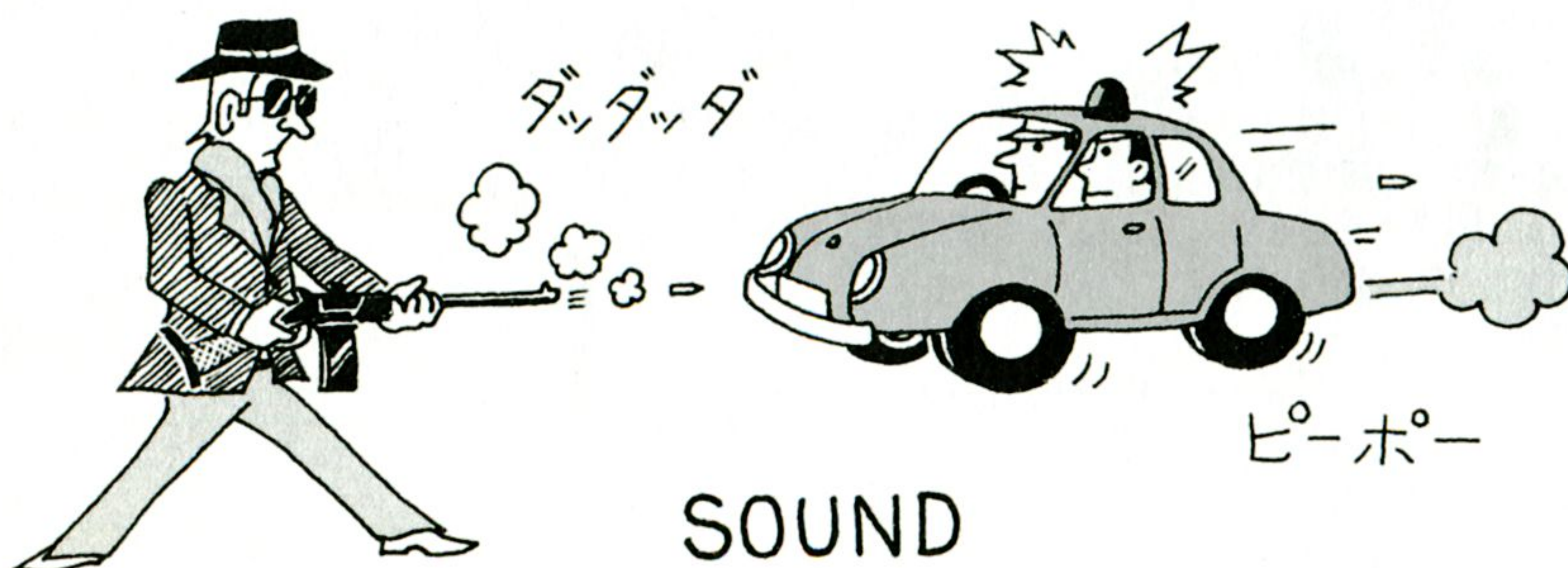
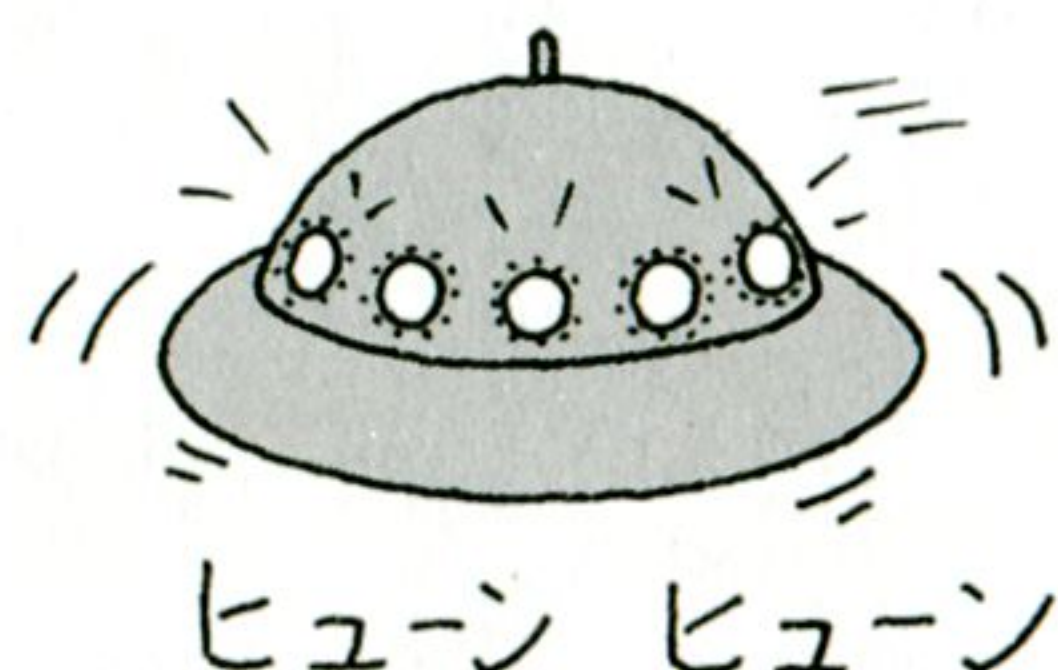
Quick BASIC

効果音のないゲームなんて

映画『スターウォーズ・帝国の逆襲』で戦闘機 XWING がデススターの表面を飛び回るシーンがありました．かっこいいですねー．まさに手に汗握るとはこのことです．でも，もしあのシーンで，音が何もしなかったらどうでしょう？急に迫力が減ってしまいます．やはり，ゲームにも効果音が必要です．そこで，今度は効果的な音の作り方について勉強しましょう．

Quick BASIC で音を発生させる命令は，PLAY(プレイ) と SOUND(サウンド) の2つです．今回はその1つ SOUND 命令を使いましょう．SOUND 命令の使い方は，次のとおりです．

SOUND_ 周波数,_時間

周波数には，38〜32767の間の数を入れてください．これ以外の数値を入れるとエラーとなります．時間は，数値が1増えるごとに約0.055秒増加します．1秒間音を出したいときには18.2を入れてください．

自分の求める音の周波数と時間を探すには，いろいろな数字を入れて試してみるのが一番良い方法です．

誠に残念ながら，PC-9801E/F/M/U などでは，SOUND. PLAY 命令で音程を変えることはできません．

●パトカー

ではここで，パトカーに似た音を作ってみましょう．プログラム PATROL. BAS を入力して，実行してみてください．

```
PATROL．BASプログラム
  FOR I = 1 TO 5
  SOUND 1000, 15
  SOUND 500, 15
  NEXT I
```

どうです，少しは似ていましたか？　エッ，少しテンポがノロすぎる．では，時間に当たる数字を少し小さくして試してみてください．どうです，うまくいきましたか？

●キカン銃

キカン銃のようにトギレ，トギレの音はどうするのでしょうか？　プログラム KIKANJUU.BAS を入れて実行させてみてください．

```
KIKANJUU．BASプログラム
  FOR J = 1 TO 15
  SOUND 38, 1
  FOR I = 1 TO 1000
  NEXT I
  NEXT J
```

音をトギレ，トギレにさせるためには，プログラム KIKANJUU.BAS を入力して 3 行目から 4 行目までのように，間に FOR NEXT のグループを入れて，時間を消費させるようにします．FOR_I_=_0_TO_? の?にあたる数字の値を大きくすればするほど，無音の時間が長くなります．

このように，SOUND 命令は，任意の周波数の音を任意の時間発生させられる特徴を持っています．

小鳥のさえずり・エンジン音・ビーム砲の音など，いろいろな音に挑戦してみてください．

Quick BASIC

音楽のないゲームなんて

●彼女の誕生日のためのバースデーソング

機種によっては，音程を変えることができず，音楽が実行できません．ゴメンナサイ．

ゲームの世界には音楽が付きものです．音楽がなければ，なんと淋しいものになってしまうことでしょう．

最近では，ゲームから音楽が独立して「光栄」のサウンドウェアシリーズでは，ほとんどのソフトウェアに対応するCDが存在します．また，『ドラクエ交響曲』のような，フルオーケストラ版も出ています．

Quick BASICで音楽を演奏してみましょう．

プログラム“BIRTHDAY. BAS”を入力して実行してみてください．

BIRTHDAY．BASプログラム

```
PLAY "T251O3C3C5D2C2F2E1C3C5D2C2G2F1 "
PLAY "C3C5>C2<A2F2E2D1O3B-3B-5A2F2G2F1"
```

どうですか，ちゃんとバースデーソングが聞こえてきましたか？

Quick Basicでは，音楽を実行することができます．これでぐっと迫力が増してきます．では，ここで音楽に関する命令を見て行きましょう．

まずPLAY命令ですが，これが音楽の実行命令です．あとの”で囲まれた中の命令を音楽命令として実行します．まずT251ですが，Tのあとの数字(32～255)は，演奏のテンポを表します．この数字が大きければ演奏速度が上がります．

T251のあとのO3ですが，Oのあとの数字(0～6)により，演奏する音のオクターブを指示します．

＞を使うと，いまのオクターブから１つ上のオクターブに，＜を使うと，１つ下のオクターブに移ることができます．以後は音の命令です．

音の高さは，次の文字で表します．

ド　レ　ミ　ファ　ソ　ラ　シ

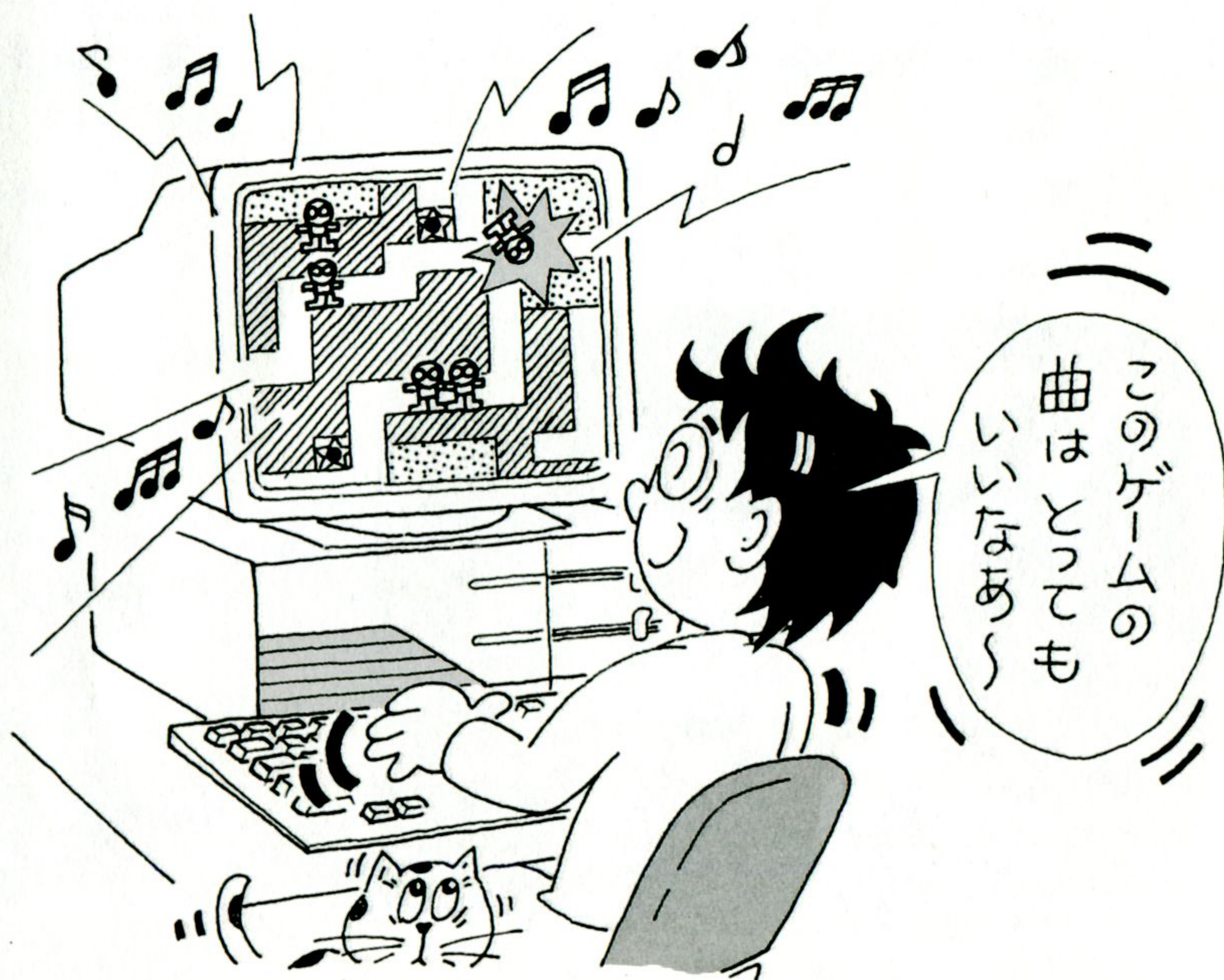
C　D　E　F　G　A　B

また，この文字の直後に－を付けると半音下がり，＋を付けると半音上がります．

各音を表す記号のあとに付く数字は，音の長さを表します．

１～64までの数字を入れます．数字の値が大きくなるほど，音の出ている時間は短くなります，音を表す文字の替わりにPを書き，その後に数字を書くと，数字によって無音の時間を作ることができます．

演奏前に，　PLAY "MB"　の命令を書いておくと，バックグラウンドで命令を行うことができます．つまり，音楽を演奏しながら，プログラムは次の処理を行うことができます．これを上手に使うと，音楽を演奏しながらゲームを行うことができ，市販のゲームにグット近づきます．

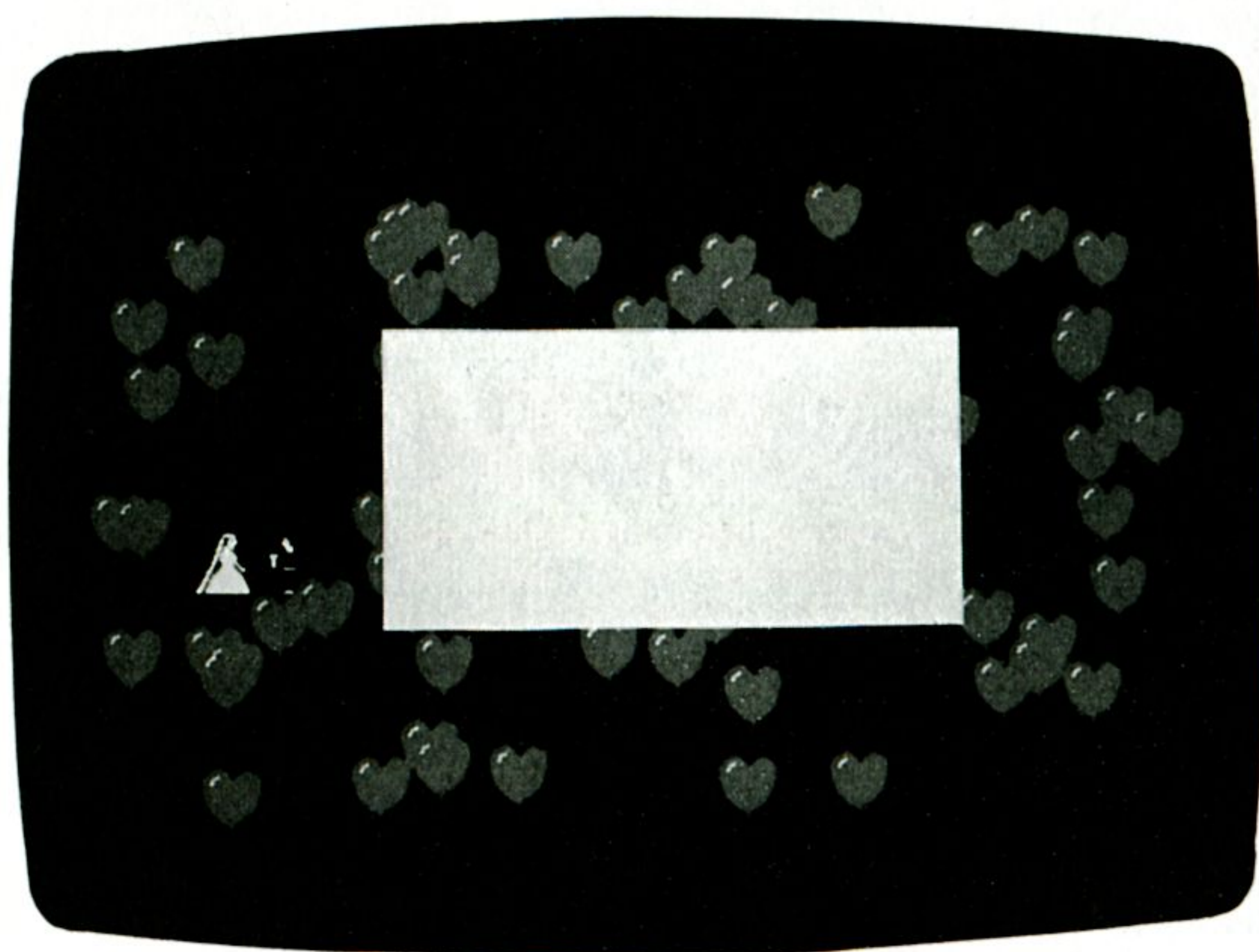
Quick BASIC

これで，彼女のハートは君のもの

★アニメーションを使ったプレゼンテーション

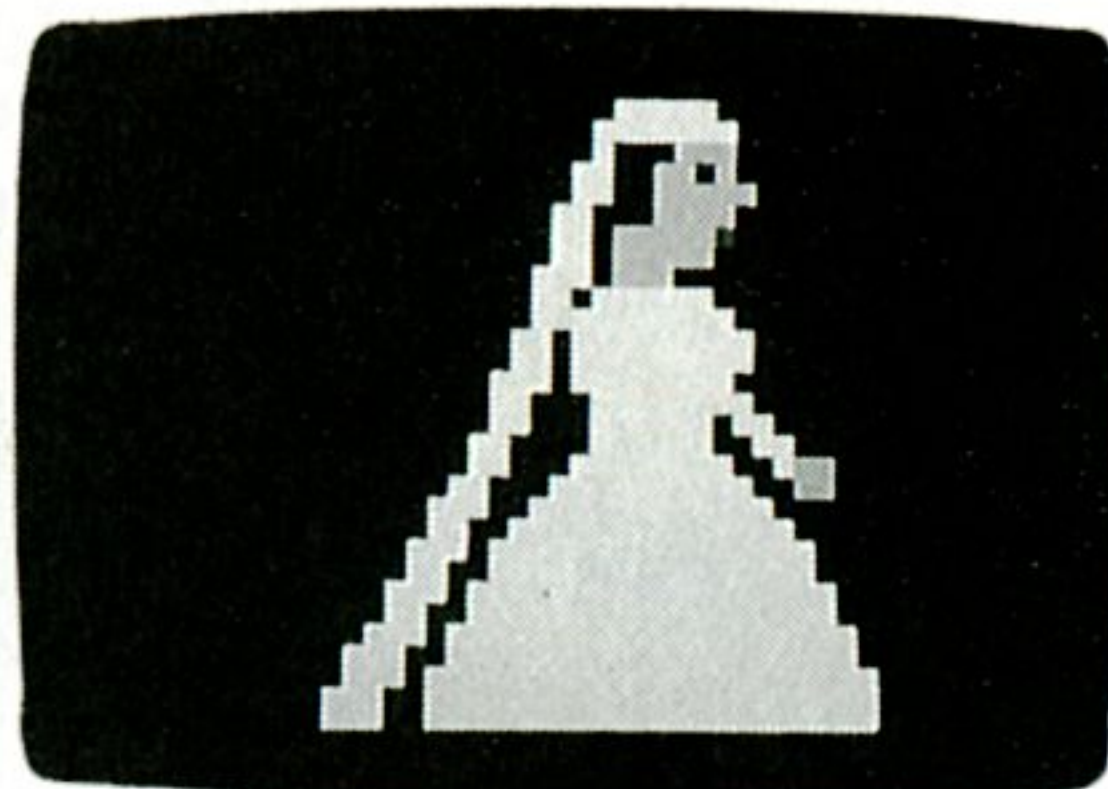
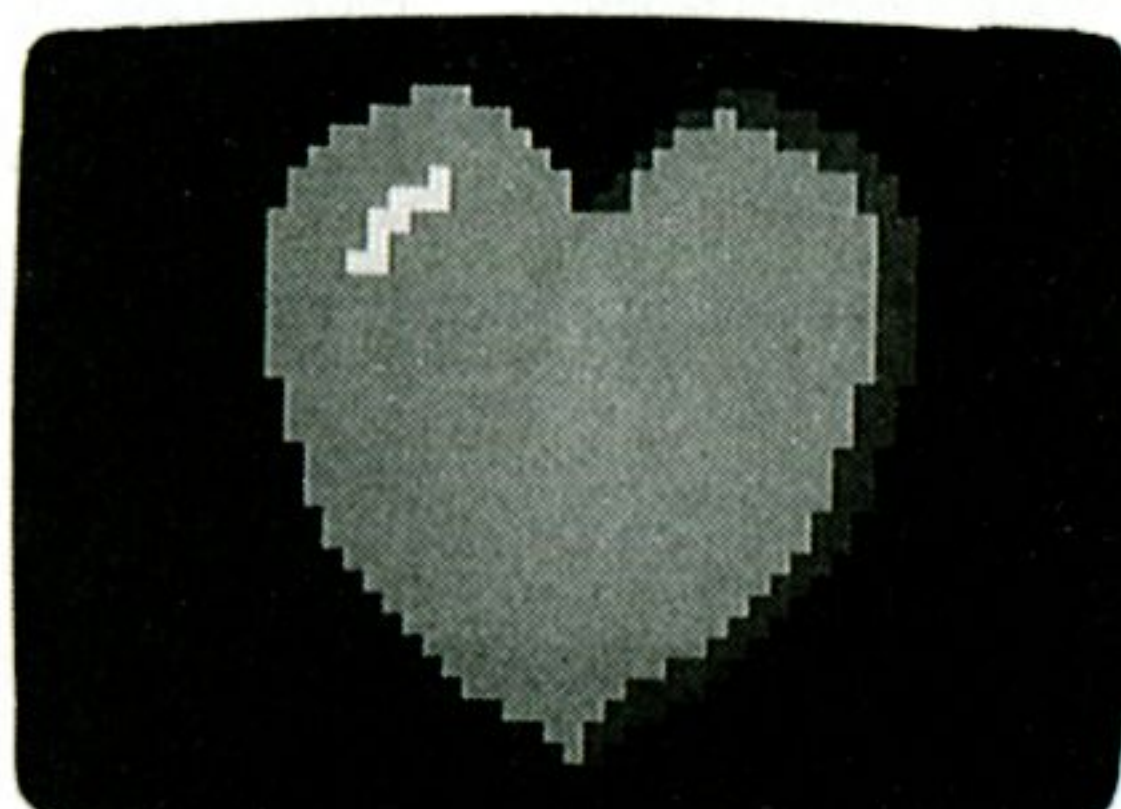
彼女への愛の告白．ウーン，難かしいですねェ．
心臓はドキドキ．口の中はカラカラ…….

世をあげて，プレゼンテーションの時代です．

いま，アメリカではパソコンを使ったプレゼンテーションが大はやりです．同じ企画や資料を見せるのでも白黒のコピーではなく，パソコンの画面上に写真やイラスト，アニメーションを上手に取り入れながら，また，音や音楽，ナレーション付きで上手に表示するのとでは，受け手の受け取り方が全然違ってきます．

パソコンの画面を，大型の液晶プロジェクタに表示できる会議室は大はやりです．そこで，私達もアニメを使ってプレゼンテーションに挑戦してみましょう．彼女が「ワァ，カワイイ！」といってくれれば大成功です．

★カラー口絵
p.4参照

『プレゼント』(愛の告白のプレゼンテーション)

★ カラー口絵 p. 4参照

BLIDE(花嫁)　MANS(花婿直立)　MANL(花婿左足前)

MANR(花婿右足前)　HART(ハート)

アニメーションの原理は簡単です．テレビや映画の絵がなぜ動いて見えるかというと，少しずつ違った静止画を順番に写し出しているからなのです．非常に早い速度でこれを行うと，人間の目にはあたかも絵が連続して動いているように見えます．

Quick BASICによるアニメーションも同じで，少しずつ違った絵を多数用意しておき，順番に表示します．違った絵をたくさん用意するのが大変ですが，キャラクタジェネレータを使って，描いた絵を呼び出し，少しだけ変えて別の絵にして違った名前で保存するという作業を繰り返すことにより用意します．チョット大変そうですけれど，彼女のハートを射止めるため，ガンバッテね．

例によって，口絵4ページ目の画面例を参考に， CGENE.BASを使って，BLIDE, MANS, MANL, MANR, HARTの絵を描いて，それぞれの名前でセーブしておいてください．

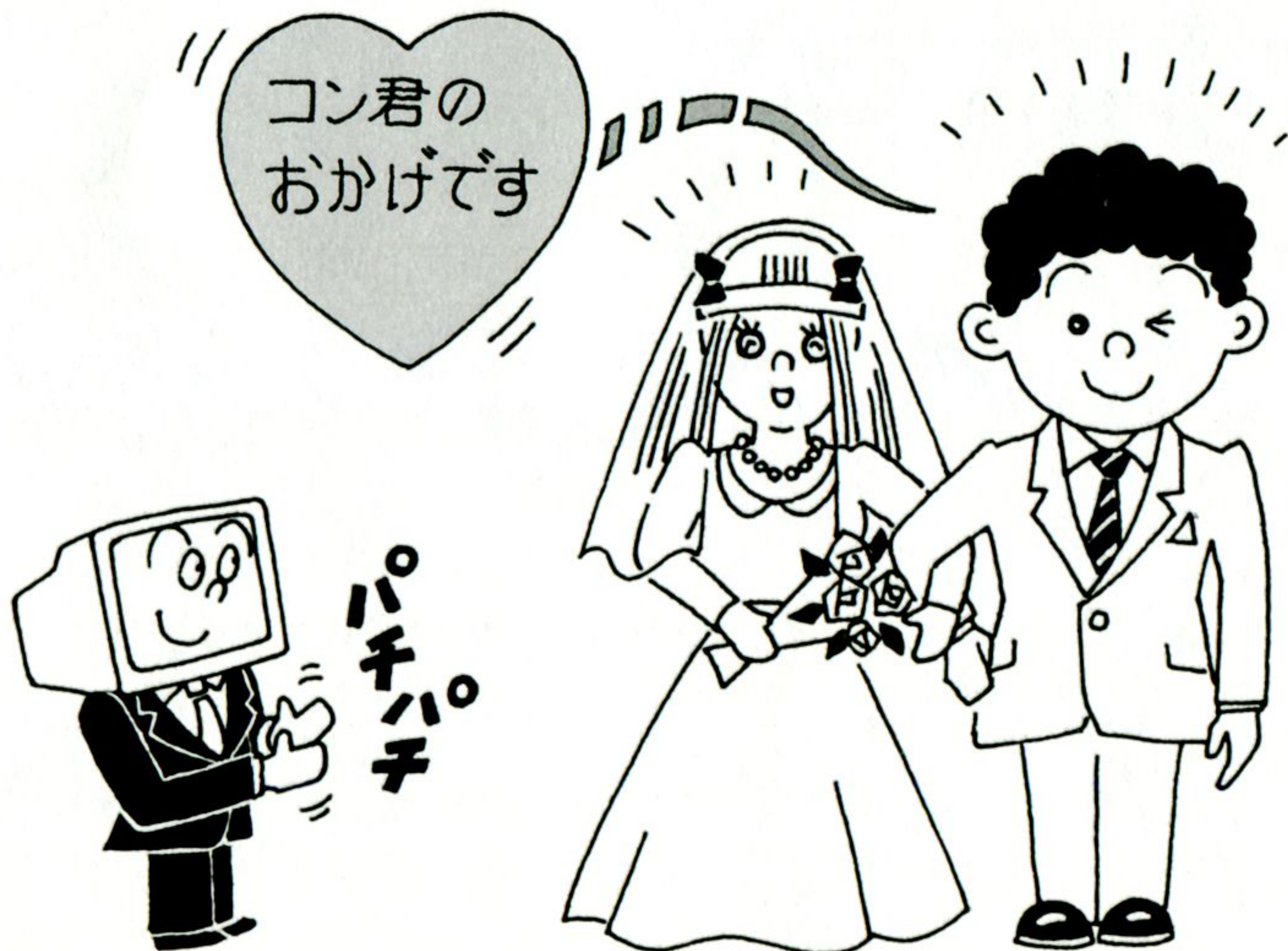
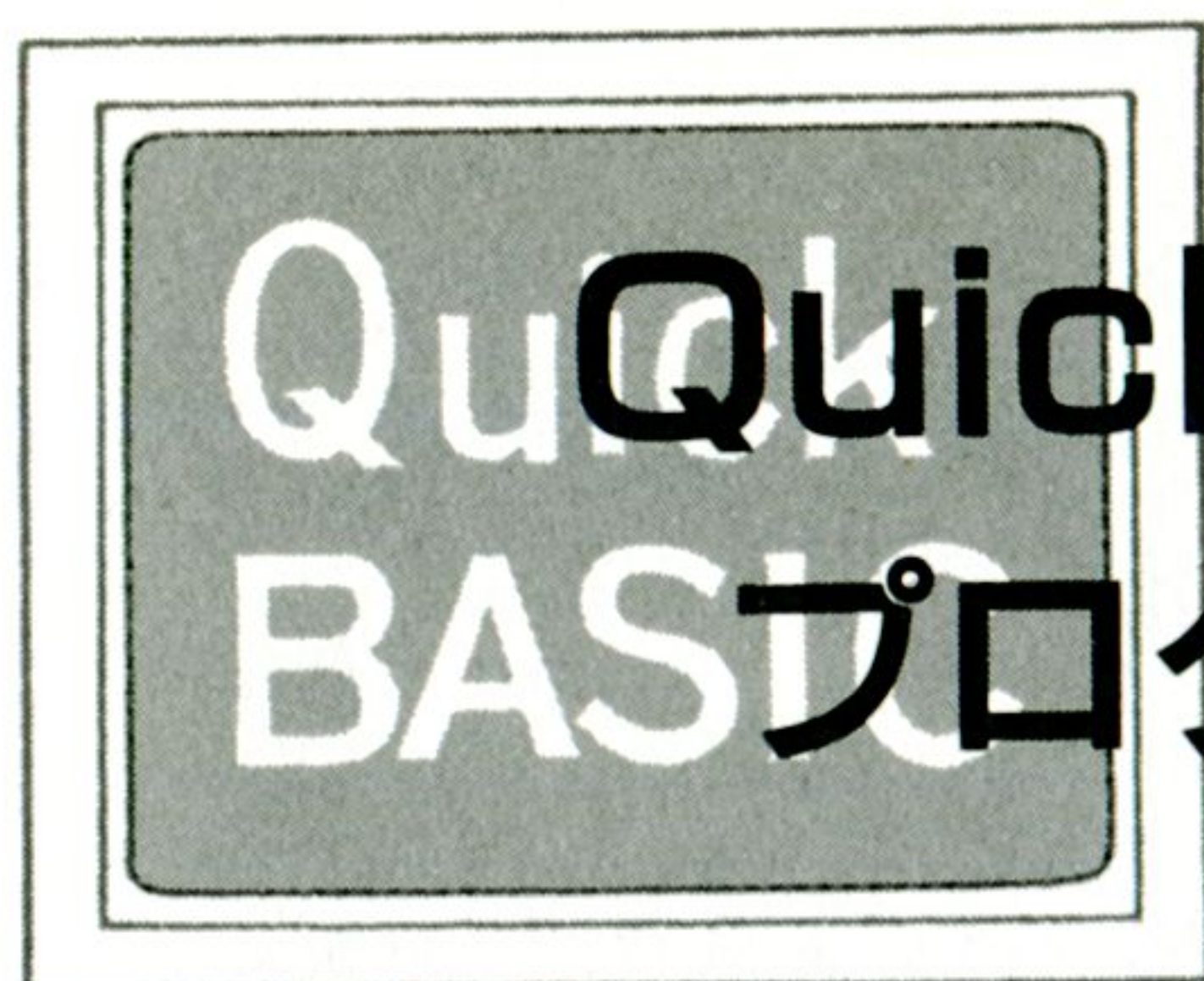
絵ができたら，次ページのプログラムPRESENT. BASをていねいに入力して保存，実行してみてください．

このプログラムでは，最後に白い四角が出てきますので，この中に彼女への愛のメッセージを書くようにプログラムを直してください．間違っても，ほかの女の子の名前なんかにしてはだめですよ！　100%フラれちゃいますからネ．では，あなたの幸運に乾杯！

ＰＲＥＳＥＮＴ．ＢＡＳプログラム

```
  SCREEN 88, 3, 1, 1
  WINDOW SCREEN (0, 0)-(639, 399)
  CLS
  MANYOKO = 500
  MANTATE = 200
  DELAY = 4000
  HARTYOKO = 100
  HARTTATE = 100
 DIM SHOUKYO%(800)
  GET (1, 1)-(32, 32), SHOUKYO%
  DIM BLIDE%(800)
  DIM MANS%(800)
  DIM MANL%(800)
  DIM MANR%(800)
  DIM HART%(800)
 DEF SEG = VARSEG(BLIDE%(0))
 OFFSET% = VARPTR(BLIDE%(0))
 BLOAD "BLIDE.GRA", OFFSET%
 PUT (100, 200), BLIDE%, PSET
 DEF SEG = VARSEG(MANS%(0))
 OFFSET% = VARPTR(MANS%(0))
 BLOAD "MANS.GRA", OFFSET%
 PUT (MANYOKO, MANTATE), MANS%, PSET
 DEF SEG = VARSEG(MANL%(0))
 OFFSET% = VARPTR(MANL%(0))
 BLOAD "MANL.GRA", OFFSET%
 DEF SEG = VARSEG(MANR%(0))
 OFFSET% = VARPTR(MANR%(0))
 BLOAD "MANR.GRA", OFFSET%
 DEF SEG = VARSEG(HART%(0))
 OFFSET% = VARPTR(HART%(0))
 BLOAD "HART.GRA", OFFSET%
 DO WHILE MANYOKO > 140
 FOR I = 1 TO DELAY
 NEXT I
 MANYOKO = MANYOKO - 10
 PUT (MANYOKO, MANTATE), MANL%, PSET
 FOR I = 1 TO DELAY
 NEXT
 MANYOKO = MANYOKO - 10
 PUT (MANYOKO, MANTATE), MANR%, PSET
 LOOP
 PUT (MANYOKO, MANTATE), MANS%, PSET
 PUT (HARTYOKO, HARTTATE), HART%, PSET
 FOR I = 0 TO 15000
 NEXT I
```

```
FOR I = 1 TO 100
    HARTYOKO = 50 + INT(RND * 550)
    HARTTATE = 20 + INT(RND * 300)
    PUT (HARTYOKO, HARTTATE), HART%, OR
        FOR J = 1 TO 300
        NEXT J
NEXT
FOR I = 0 TO 15000
NEXT I
LINE (200, 100)-(500, 250), 7, BF
```

（プログラムを完全に入力したにもかかわらず動作しない場合は，11ページの囲み，および90ページのコラムを参考にしてみてください）

Quick BASIC は プログラム言語の英語？

いまや英語は，世界の標準語です．英語をしゃべれば世界中いろいろなところで困りません．また，飛行機の管制から科学論文まで，英語でなければ困ってしまう分野がいっぱいあります．アメリカ西海岸では，英語と日本語，両方がきちんと話せる人は引っぱりだこです．

★君のプログラムがIBM PC上で全世界どこでも動く

実は，Quick BASICは，プログラム言語の英語なのです．

エッ，単語が全部英語だから英語に決まっているって？　そういう意味ではありません．

Quick BASICで作られたプログラムは，まったく改造なしか，またはほんの少しの改造で，非常に多くのパソコン上で動きます．日本のパソコンでも，PC-9801だけでなく，富士通のFMRシリーズ，FM TOWNS，IBM55，AXなど，そのマシン用のQuick BASICがあります．それを買えば，プログラムの移植も簡単です．

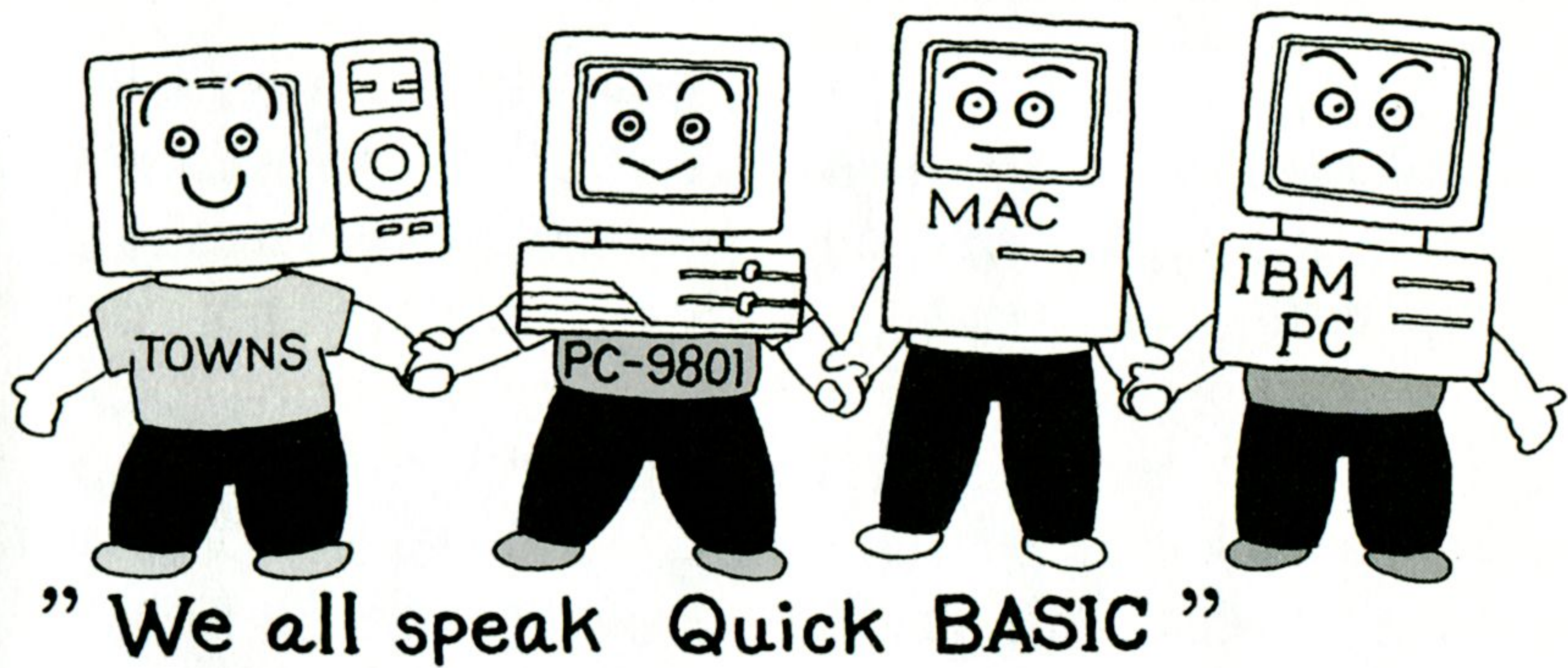
それより何より一番すごいのは，世界中でもっとも使われているパソコンのIBM　PCとそのコンパチブル機でも同じように最小限の努力で動きます(Quick BASICはもともとIBM PCのために作られた言語ですから，当たり前といえば，当たり前ですが)．

いまも，私の机の上では，IBM PCとPC-9801の2台のマシン上でプログラムがいったりきたりしています．

ほかの言語でもIBM PCとPC-9801で共通のものもあるのですが，グラフィック命令がかなり違ったりして，書き換えるのが大変です．もちろん，アップルのマッキントッシュ用のQuick BASICもあります．

ぜひ，すばらしいスーパーゲームを作って，日本だけでなく世界に売りまくってください．アメリカでも“Nobunaga's Ambition”(信長の野望)などの光栄のゲームや“テグサー”などが売れています．

日本IBMがDOS/Vなどを売り出して，世界中のIBMで日本語を使えるようになってきたり，セガがテラというメガドライブとIBM両方になれるマシンを作るなど，IBM PCが普及するにつれ，簡単に移植できるQuick BASICの価値がますます高まります．

Quick BASIC

スーパーコンピュータもわかったゾ!

スーパーコンピュータって知ってますか？　計算速度がメチャクチャ速くて，天気の数値予報をしたり，車や飛行機の周りの空気の流れを計算して型を決めたり．果ては，車をどう作れば，どれぐらいの強度になるか計算したりできます．筆者の弟は，車を作る仕事をしていますが，昔はとりあえず作ってみては壁にぶつけて壊れ方を調べ，部品の厚さや強度を変えて，また作ってはぶつけるということを何回か繰り返し，やっと設計を決めていたそうです．いまでは，まずコンピュータの中で架空の自動車を壁にぶつける計算をして，その計算によって部品を決め，実際に作って計算を確かめる操作は１回で済むというように，時間，物，金，すべてにわたってものすごくムダが省けるようになったそうです．

スペースシャトルや超音速機などは，スーパーコンピュータがなければ，作ることができません．

まさに，これからの世の中，「スーパーコンピュータを制するものが世界を制する」といっても過言ではありません．

ところで，このスーパーコンピュータは，ほとんどがFORTRAN(フォートラン)という言語によってプログラムを速く動かすように作られています．科学技術計算の世界では，言語はFORTRANが主流なのです．過去に作られたFORTRANのプログラムの資産は，作り直すとしたら天文学的な時間とパワーを必要とします．

「なぜQuick BASICの本に，FORTRANの話が出てくるのか？」ですって……．

実は，ここまでQuick BASICを勉強してきたあなたは，このFORTRANの世界にあと１歩のところまで来ているのです．FORTRANとBASICは驚くほど良く似ています．最初にFORTRANを学び始めたとき，あんまり似ているのにビックリしてしまいました．

マア，BASICを作ったアメリカ・ダートマス大学のケメニー(J. Kemeny)とカーツ(T. Kurtz)の2人が，FORTRANの前に学生に簡単にわかるようにと，まねして作ったのだから，当然といえば当然なのですが…….

アメリカでは，国家が『スーパーコンピュータの力が，将来の国の力だ』ということをよく認識しています.

毎年，全国からおもしろいテーマを持った数学やコンピュータが大好きな高校生を何人も集めます．そして，国の費用で，スーパーコンピュータを使わせてくれるのです.

もちろん，1人ではプログラムの作り方や機械の操作がわからないと困るので，専門の係の人がついてくれます．人種などにもこだわらず，この間は日本人の留学生が選ばれていました．このスーパーコンピュータの1時間の使用料が，何十〜何百万円というしろものですから，いかにすごいかがわかります．アア，ウラヤマシイ!

日本にも早くこういう時代が来るといいのですが…….

ここまでプログラムの力を付けてきたあなた，ぜひ，スーパーコンピュータ目指して頑張ってください.

コラム

〔秘話〕ディップスイッチ

昔々，ある通信ソフトとモデムを使って，通信をしようとしましたが，どうもうまくいきません．どうやらディップスイッチがあやしいらしく，マニュアル片手に1がON，2がOFFとやっていましたが全然うまくいきません．これはもうだれかに助けてもらう以外はないと，NECに電話して，係の人に1がONで…と，悪戦苦闘30分，そのとき係の人が一言，「お客様，このディップスイッチ下がONですけれど，ヒョットして…」

・・・

グット沈黙のあとで，「ゴメンナサイ」を繰り返して電話を切った私でした.

以来，絶対にディップスイッチの表示は，ON / OFFではなく，上と下がいいと信じています.

Quick BASIC

これであなたも一流プログラマー

どうです．ここまでやってきて，コンピュータのプログラムに必要な基本的なことは，ほとんど勉強してきました．プログラム作りも，ゲーム作りも意外と簡単だったと思いませんか？

あとは，“自分がなにをやりたいのか”というアイデアとコンセプトの問題です．アイデアとコンセプトさえしっかりしていれば，それを実際のプログラムに書いて行くやり方は必ず見つかるものです．

もうあなたも一流プログラマーの仲間入りができました．あとは，ゲームでも技術計算でも，自分のやりたい分野で思いっきり腕をふるってください．

★ Have a nice day !

ここまで，できるだけ楽しくプログラミングの勉強ができるようにと書いて来ましたが，いかがでしたか．

コンピュータの世界が楽しいのは，自分が命令した通りに動いてくれることです．プログラムの中にバグ(間違い)があって動かないときでも，それを探し出して動くようにするのは，まるで推理小説の中で，自分がシャーロックホームズかポワロになった気分です．おまけに結果はすごくはっきり出て来るのですから……．

プログラムを作る上で一番大切なのは，考え方です．論理の流れがしっかりしていれば，どんなプログラム言語を使っても，固有の命令に置き換えるだけで済みます．みなさんは，この本の中で十分にその力が付いているといえるでしょう．

あとは，自分のやりたいことを楽しみながら挑戦してください．

では，またいつか，どこかでお目にかかれる日を楽しみにしています．

コラム

おまけ(MS-DOS　隠しコマンド)

PC-9801とIBM　PCでデータを簡単に行ききさせる方法(5インチ2HD用)を述べます.

MS-DOSの一部のFORMAT(フォーマット)コマンドには，隠しコマンドがあります. MS-DOSのシステムディスクをAドライブに入れ，新しい2HD版のディスクをBドライブに入れてください.

この状態で，FORMAT_B:/5 ↵ と打ち込んでください. こうやってFORMATした5インチ2 HDのディスクは，ともに2HDの読めるPC-9801とIBM PCどちらからでも読み書きできます.

PC-9801で書き込んだQuick BASICで作ったプログラムを，テキスト型式で，このいま作ったディスクに保存します.

このディスクをIBM PCの5インチ(2HDの読めるタイプ)に入れてIBM PC用のQuick BASICで読み出すと，なんなくデータの移動に成功します. ヘタなコンバージョンソフトや，2台をつなぐリンクプログラムやLAN(ラン)(ローカルエリアネットワーク)などいりません. また，キャラクタジェネレータでセーブした絵も，そのまま使えます.

IBM PCとPC-9801の間のデータ交換は，FORMAT/5のディスクを使えばできるのは，いま述べたとおりですが，このとき，必ずテキスト形式でSAVE(保存)してください. テキスト形式でのSAVE(保存)は，ファイルを呼び出し”名前を変えて保存……”を選びます. そのとき，右側にファイル形式をQuick BASIC標準ファイルかT/テキストファイルにするか出ているので，T/テキストファイル側のDOTをマウスでクリックするか，TABキーと矢印キーで選んで，その後，確認をクリックするかリターンキーを押してください. テキストファイル形式でSAVEしないと互いにファイルが読めません.

おことわり：この方法は，正規にサポートされていないので，将来的にやめになるかも知れません. また，DOSのバージョンや，機械の相性によっては，動かないかも知れません.(「オイ，動かねえ，ドウシテクレル」とすごまれても，当局は一切関知しませんのであしからず！)

索　引

（アルファベット順）

（五十音順）

索　引